2012-06

Optimized Landing of Autonomous Unmanned Aerial Vehicle Swarms

Dono, Thomas F.

Monterey, California. Naval Postgraduate School

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

OPTIMIZED LANDING OF AUTONOMOUS UNMANNED AERIAL VEHICLE SWARMS

by

Thomas F. Dono

June 2012

Thesis Advisor:	Dr. Timothy Chung
Second Reader:	Major Chad Seagren, Ph.D.

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From — To)*
7–6–2012	Master's Thesis	2010-06-01—2012-05-31

4. TITLE AND SUBTITLE

Optimized Landing of Autonomous Unmanned Aerial Vehicle Swarms

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Thomas F. Dono

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Postgraduate School
Monterey, CA 93943

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Department of the Navy

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited

13. SUPPLEMENTARY NOTES

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

14. ABSTRACT

This research explores a future concept requiring the efficient and safe, landing and recovery of a swarm of unmanned aerial vehicles (UAVs). The presented work involves the use of an overarching (centralized) airspace optimization model, formulated analytically as a network-based model with side constraints describing a time-expanded network model of the terminal airspace in which the UAVs navigate to one or more (possibly moving) landing zones. This model generates optimal paths in a centralized manner such that the UAVs are properly sequenced into the landing areas. The network-based model is "grown" using agent based simulation with simple flocking rules. The resulting solution is compared to another agent-based model which uses similar avoidance rules for the landing of these UAVs, exploring the benefit of distributed computation and decision-making characteristic of swarming models. Relevant measures of performance include, e.g., the total time necessary to land the swarm. Extensive simulation studies and sensitivity analyses are conducted to demonstrate the relative effectiveness of the proposed approaches.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			19b. TELEPHONE NUMBER *(include area code)*
Unclassified	Unclassified	Unclassified	UU	83	

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

OPTIMIZED LANDING OF AUTONOMOUS UNMANNED AERIAL VEHICLE SWARMS

Thomas F. Dono
Major, United States Marine Corps
B.S., Psychology, Virginia Tech, 1999

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS ANALYSIS

from the

NAVAL POSTGRADUATE SCHOOL
June 2012

Author: Thomas F. Dono

Approved by: Dr. Timothy Chung
Thesis Advisor

Major Chad Seagren, Ph.D.
Second Reader

Dr. Robert Dell
Chair, Department of Operations Research

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

This research explores a future concept requiring the efficient and safe, landing and recovery of a swarm of unmanned aerial vehicles (UAVs). The presented work involves the use of an overarching (centralized) airspace optimization model, formulated analytically as a network-based model with side constraints describing a time-expanded network model of the terminal airspace in which the UAVs navigate to one or more (possibly moving) landing zones. This model generates optimal paths in a centralized manner such that the UAVs are properly sequenced into the landing areas. The network-based model is "grown" using agent based simulation with simple flocking rules. The resulting solution is compared to another agent-based model which uses similar avoidance rules for the landing of these UAVs, exploring the benefit of distributed computation and decision-making characteristic of swarming models. Relevant measures of performance include, e.g., the total time necessary to land the swarm. Extensive simulation studies and sensitivity analyses are conducted to demonstrate the relative effectiveness of the proposed approaches.

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

List of Figures

List of Tables

THIS PAGE INTENTIONALLY LEFT BLANK

List of Acronyms and Abbreviations

ABM Agent Based Model

FAA Federal Aviation Administration

ISR Intelligence, Surveillance and Reconnaissance

LZ Landing Zone

LZs Landing Zones

NAS National Airspace

NPS Naval Postgraduate School

UAS Unmanned Aerial System

UAV Unmanned Aerial Vehicle

UAVs Unmanned Aerial Vehicles

THIS PAGE INTENTIONALLY LEFT BLANK

Executive Summary

Unmanned Aerial Vehicles (UAVs) are increasingly utilized in the operational environment today, by both our friends and our enemies. As this technology grows so does the requirement for increased automation of these systems, ultimately enabling one operator to control tens of UAVs, if not more. One area of research is the automation of airspace management in either the terminal environment or in a target area. The UAV swarm would have to be assigned a destination and negotiate the airspace to reach its intended goal in a timely and efficient manner while avoiding collision from other UAVs.

This thesis explores this problem utilizing two different approaches. The first approach is through the use of a centralized control solution. This is formulated using a network-based model that ultimately outputs control paths for the individual UAVs as they sequence for landing. The second approach is the use of a decentralized solution. In this approach we use an agent based model, implementing flocking heuristics to create the desired behaviors.

The network-based model is formulated as a time-expanded network model with side constraints. The formulation can easily become too large to solve in a traditional way. We create several algorithms that are able to significantly cut nodes away from the network, ultimately making the model solvable for operationally relevant airspaces. Some of these approaches include the use of convex hulls and agent based techniques to select and cut nodes. This has the effect of making the model tractable by today's computational equipment and solutions readily available to the operator.

The agent based model uses basic flocking heuristics developed by Reynolds [1]. These heuristics utilize the following rules:

- **Flock Centering**: Desire for agents to move toward the center of mass of the other agents.
- **Collision Avoidance**: Repulsive force causing the agents to want to be apart from each other.
- **Velocity Matching**: This force causes the agents to want to move in the same direction.
- **Tendency**: This force draws the agents to a goal or destination.

These rules provide a basic structure for the behaviors of our UAVs as they traverse the airspace.

These two approaches are compared to one another ultimately realizing that the agent based,

decentralized approach to swarm control is almost as efficient as the centralized approach, at fraction of the computational cost of the centralized approach. This leads to the conclusion that Unmanned Aerial Vehicle (UAV) swarms may benefit from the use of an intelligent combination of both a centralized control and decentralized control solution that can take advantage of the strengths of both these methods.

Optimization models, specifically certain network formulations, can benefit from the use of agent based or Monte Carlo simulations as a part of their formulation. We show that it can significantly reduce computational requirements and even have the effect of making intractable or overly difficult problems more feasible and easier to solve.

Agent based models, alternatively, can and should use optimization as a benchmark for the development of complex behaviors and determining simulated system performance. The optimal solution provided by these models can give a measure of effectiveness to a simulation modeler as they strive to understand the nature of a system.

Acknowledgements

I would first like to thank my lovely wife Kristen and my daughters Isabella and Luciana. They have been nothing but rock of support through my graduate school education and my military career in general. Without their love and caring I could not have done this work.

I would like to thank Dr. Timothy Chung, my thesis advisor, for giving my enough rope to hang myself then reining me in just before I did. Through him I have been able to learn much more than I possibly could have myself. He is an outstanding mentor and professional, and again this thesis would not have been possible without his help and vast knowledge.

Professor Carlyle has been invaluable with his help in model formulation and programming in GAMS. His instruction and aid has been instrumental in completing this project. Thanks to his incredible knowledge of GAMS and programming I was able to get this model off the ground.

I thank Dr. Ned Dimitrov for introducing me to network optimization and giving me the inspiration to pursue this line of thought.

Finally I would like to thank Sudeep Pillia who was gracious enough to allow use his code for this thesis, saving me hours of coding and experimentation. Because of Sudeep we were able to get the agent based model up and running in a few hours.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 1: INTRODUCTION

1.1 Background

"The fiercest serpent may be overcome by a swarm of ants."
- Isoroku Yamamoto

UAVs are growing in popularity. Currently, UAVs are used for missions ranging from Intelligence, Surveillance and Reconnaissance (ISR) collection to target prosecution. In the current operational environment, at least one human operator is required to control a single UAV during flight. As this technology grows there will eventually be tens of UAVs controlled by a single human operator. These large groups of UAVs, or *swarms*, will be used for missions ranging from attacking targets to the defense of high value assets. The level of autonomy must increase appropriately in order to reduce operator workload and still allow for mission effectiveness and completion.

In addition, current and future potential adversaries of the United States are developing such unmanned systems capabilities for offensive attack against friendly targets. A possible concept of operations may be to arm them with high explosive warheads and to send them out to attack targets, similar to the kamikazes used in World War II. These "swarms" of inexpensive unmanned vehicles have an advantage over more conventional weapons like missiles and rockets; they can be launched from remote locations hundreds of miles away from their intended target, then loiter at a predetermined location, search for targets, and attack those targets. The relatively low cost of these weapons allows our enemies to procure large numbers of them to swarm attack a single target, overwhelming its defensive systems, and thus becoming a cheap and highly effective method to attack todays most sophisticated and expensive warships.

The concept of swarming is not novel. The natural world uses swarming as an effective method for taking down larger prey and enemies, as this behavior is indicative of bees, ants, and many others [2]. Humans have used swarm tactics to attack their opponents for some time as well. The Mongols attacks throughout Asia, the Japanese swarms of kamikaze airplanes attacks on the Allied Fleet, and the Romans used swarms of dogs to break up enemy formations and decrease their morale [2, 3].

"Swarms" are nominally defined differently from "teams." A swarm is guided by a common

objective, but often with little or no centralized coordination. The very nature of a true swarm is characterized by its uncoordinated, inexpensive and overwhelming traits, e.g., the worker bee is cheap when compared to the hive or the queen. In the context of UAV swarms, it must adhere to these three properties to be considered a true swarm [4]. Application of the "uncoordinated" element can be reduced to distributed decision making, where the UAV needs to "decide" without explicit coordination with the other UAVs. To be considered inexpensive, one considers both the cost of materials and also computational complexity. Finally, swarms seek to overwhelm the adversary with two main effects. The swarm needs to saturate and overcome the defensive systems of the target, and it can have an intimidating psychological effect on the enemy, inducing fear and destroying morale. War is not, after all, a conflict between mechanical systems, but between human wills [5].

In contrast, a "team" is often explicitly characterized by its coordinated and controlled capabilities. Teams usually have a leader or some element of a command structure. In nature, teams are seen in wolves or pods of orca whales, where smaller groups work in a coordinated manner to achieve a specific goal. In the case of teams, each team member, let alone the team leader, often plays a critical role, and the loss of certain individuals could be detrimental to the success of the mission. This latter architecture represents the current paradigm for most military organizations.

Thus, the use of swarms can and should be explored by the United States and military allies to be used against its enemies, both in offensive and defensive operational contexts [3]. The swarming tactic should be imagined as a large force of small UAVs attacking a target, similar to bees attacking a threat to their hive. A swarm of UAVs can flood an airspace around a target area and attack this target near simultaneously, and overwhelmingly. The target then has to choose between UAVs, while realizing that while it is engaging it is getting attacked and receiving damage from other UAVs that it can not handle.

The employment of such swarms encompasses a wide range of technological and operational challenges relevant to multiple mission areas. For example, the problem of assigning assets to simultaneously strike a number of targets can also be related to the problem of sequencing the efficient autonomous landing of UAVs. Consider the ongoing development of the Navy's X-47B UAV, which has been designed for carrier fleet operations with the ability to attack targets by delivering munitions like conventional manned aircraft (e.g., F/A-18). In addition, this UAV has recently demonstrated its ability to successfully land autonomously in test environments as a precursor to carrier deck landings [6, 7] In this case the group of X-47B aircraft would have to organize and land in accordance to the capacity and safety of the landing platform. For example, a carrier could only accept one X-47B to land on its

deck every ten seconds. In the context of this thesis, the X-47B swarm would self-organize in such a manner that would allow for optimization of this landing sequence. Given this additional constraint to schedule individual arrivals of assets (versus simultaneous arrivals), one can view the optimization of the strike assignment as a relaxation of this optimized landing problem. For this reason, in this thesis, we explore the landing problem in depth, recognizing that it can be directly translated to the strike problem. This connection will be revisited in later sections once the general formulation has been presented.

1.2 Literature Review

The problem of UAV landing is seen in several parts during the focus of this literature review. The first part is airspace management. We are controlling a large number of air vehicles, in a finite airspace, around one or multiple landing zones. This is not much different from what the Federal Aviation Administration (FAA) does on a daily basis around major airports. The second part is simple flocking, since the swarms act in much the same way as a flock of birds or a school of fish. Airspace management is the deconfliction of traffic in a macro sense, where the flocking is a deconfliction in the micro sense.

Since the boom of air travel in the 1970s, researchers have been working to better optimize the arrival process of aircraft into busy airports. For civilian air traffic systems, the problem is a matter of fuel and cost. There has been a plethora of research done on efficiently landing commercial air traffic around busy airports. The UAV landing scenario could be visualized as a 15 to 20 minute time slice of these busy airports, receiving a large influx of air traffic in a short period of time, follow by times of low activity.

Peterson, Dimitris and Amedeo [8] look at deterministic models for describing the nature of the air traffic around Dallas Fort Worth International Airport. They use discrete time Markov and semi-Markov chains (with discrete 15 minute intervals) to determine the expected delays, and also include six different state spaces, and an arrival process that is either time-varying Poisson or deterministic to capture heavy and light traffic periods. Their model proves to be useful in describing and analyzing the delays at the airport and is valid against real data collected. This model does have its drawbacks, though, since it is computationally expensive. If a similar method were to be used for our problem model, it would require significant scaling to have a much smaller time step to determine baseline performance of our system.

Bolender and Slater [9] similarly use the $M/D/2$ and $M/M/2$ queues to evaluate the performance of multiple runway operations to determine if they are performing near or at their optimal value. They determine that most airports are not running at maximum capacity and

that the presented models are useful in predicting arrival performance of airports looking to expand runways. Saraf and Slater [10] developed an Eulerian circuit-based model used for real time optimization of arrival scheduling. The model determines the best scheduling sequence of aircraft in the current state space. Their program is able to land planes up to 25% more efficiently. Boesel [11] used an object-oriented Monte Carlo simulation model that assesses the effect of a new FAA policy on the arrival traffic at Detroit-Wayne County Airport. Their model uses individual objects for the aircraft, where each aircraft object must travel along pre-designated tracks called links. The study compares the effect of new links being used by the FAA in Detroit-Wayne Airport (DTW). They are able to evaluate the changes made to the airport's airspace routing and improve efficiency and capacity of the area. These queuing type models are not considered in this thesis since the swarm is arriving in the airspace at a deterministic time, and is in the airspace for relatively shorter periods of time in contrast to the daily operations of airport management. These models also only explain airports and airspace capacity for scheduling of aircraft, but do not find efficient paths in the airspace that are being navigated by the aircraft themselves.

Brooker [12] suggests that the use of 4D (4th dimension being time) modeling and scheduling with the aid of software will decrease arrival times and delays greatly. The 4D modeling and arrivals will not queue aircraft, but start scheduling their arrival hundreds of miles away, even prior to their take-off, thereby enabling the aircraft to arrive on time, with no holding and only small flight modifications. This work differs from the traditional priority queuing systems and $M/G/1$ queues that are often used to predict performance in airport arrival systems. This approach, unlike the previous related works, can be translated into the landing of UAV swarms, having them adjust their flight paths and profiles appropriately to enter directly into final approach with no holding or delay. More appropriately, the delay is absorbed by the time in transit and set up to approach. This helps realize the concept of UAV swarms by utilizing open airspace to timing and spacing vice circular holding patterns more traditionally used in the aviation community.

The use of network models is not new to the study of this problem. This is because the use of specialized algorithms and methods, such as Dijkstra's shortest path and network simplex [13], have greatly increased the efficiency of computation for these types of problems. In a network model the problem is broken down into a series of nodes and edges that each have their own special values and characteristics. These methods are then applied to the "graph" to gain a solution. Dell'Olmo and Lulli [14] explore network models for this problem and try to include what they refer to as "Free Flight Capacity." This is not entirely true, however, since the nodes and edges all are defined within the airspace structure as defined fixes and routes. "Defined" is a term used by the FAA to denote a point in space that has been assigned

a name and significance. In this work, they use a simplified true free space model that is both free flight and dynamic (i.e., having time as a component). This is the model that is the primary basis for the model that we explore. The K-King model uses a chess board in which a king can move in any direction on the board but only one space at a time. The king is then limited by rules or constraints, for example, a deconfliction constraint, in which he is not able to get too close to a fellow king. The kings then need to go from their current position to a designated corner of the chessboard to land [15].

Agent-based models (ABMs) have also been used for many years and in many fields. The use of these simulation models have given us great insight into many natural phenomena behaviors, like flocking birds and ant organization [16]. When expanded into larger more complex models, such models can give us insight into systems like macro-economics and combat.

Conway [17] developed an agent-based model to mimic the air traffic control systems utilized by many major airports to observe the behavior of the system. She concluded that this was a good tool for analyzing the behaviors of air traffic in the terminal environment. This type of modeling, along with the mathematical models mentioned above, could prove useful in gaining intuition on the nature of our problem. Since we are developing behaviors for deployment on actual autonomous aerial systems, agent-based models are a great tool for creating autonomous and adaptive behaviors in these systems that benefit from decentralized capabilities.

Agent-based models are also used to explain a behavior called flocking. This is particularly interesting because it is the exact behavior that is desired in UAV swarms while in flight. Olfati-Saber and Murray [18] describes several algorithms that have been used for the flocking of agents in both free and restricted space, where free space reflects the absence of obstacles in the environment. Olfati-Saber [19] later expanded their algorithms for obstacle avoidance to be more efficient and to get better performance from their agents. Yu, Jain, and Shen [20] expand on the flocking and obstacle avoidance behaviors with the use of "fuzzy-logic" with good results, having the agents form up faster with better speed and distance control than previous algorithms. Both approaches can be used for the flocking of our UAV agents in the initial phase of our experiments. All of the techniques described above are useful for en route and engagement flight profiles of swarms, since they provide a behavior similar to swarms of locusts or schools of fish. However, in landing UAVs we will require behavior that is substantially more rigid and sequenced to allow for effective and safe recovery.

1.3 Concept of Operations

UAV swarms can and will be used in a variety of operations in the near future. Missions including both defensive and offensive operations, specifically those that autonomous or semi-autonomous vehicles can perform, are often limited only by the imagination and ingenuity of both ourselves and our enemies. Some of these types of missions could include:

- Defense of infrastructure or high value assets: UAV swarms could serve as an initial line of defense against enemy UAV saturation attacks. In addition to providing this active layered defense, such UAV swarms could also conduct Combat Air Patrols to look for and target enemy UAVs as they try to penetrate our defensive airspace.
- Intelligence , Surveillance and Reconnaissance (ISR): UAV swarms could be equipped with heterogeneous sensor payload arrays, that when combined and synchronized could produce a more in-depth, multi-dimensional, multi-spectral picture of the battlespace.
- Armed reconnaissance: Beyond ISR, the platform now becomes a hunter and a killer. UAV swarms can be sent forward of the friendly line looking for targets of opportunity. Once found, those targets can then be prosecuted by the same UAV that found the target.
- Aerial interdiction: UAV swarms can be deployed to fly deep into enemy airspace, overwhelming the enemy's integrated air defense systems and/or destroying a strategic target well inland of the fighting.
- Electronic warfare: As jammers and communication technology become more sophisticated, the need for strict noise jamming is becoming obsolete. Swarming UAVs carrying more frequency-agile jammer technology could be used to disrupt and possibly deny enemy communications for extended duration and spatial coverage.

1.4 Scope and Assumptions

1.4.1 Scope of this Thesis

This thesis will concentrate on addressing the research questions below:

- What is the (near-)optimal path for each UAV in the system to attack a given target or land at a given landing zone?
- What is the trade off in terms of time and performance between centralized and decentralized control on routing a UAV swarm?

In order to address the former, the objective function the *time required for all UAVs to reach their destinations*. An optimal path minimizes this objective function by choice of trajectory

through the discretized airspace, while obeying constraints on flight speed, collision avoidance, and other relevant limitations. We explore various methods to improve the operational realism of the problem under limits in computational capacity.

The latter investigation includes the development and comparison of the optimal path solution and that is based on an agent-based heuristic. The value of this evaluation is to identify the benefits and penalties of using a centralized optimization model, whose solution is disseminated to each and every UAV versus endowing the UAV swarm elements with sufficient capability to compute their own paths in a decentralized (but sub-optimal) manner.

1.4.2 Assumptions

A number of assumptions further limit the scope of this thesis and apply to either or both of the network-based optimization and agent-based models investigated in this thesis.

- Space and time are discrete
- UAVs are homogeneous, i.e., they possess identical capabilities and flight characteristics
- UAVs are able to localize themselves perfectly (e.g., GPS) and their neighboring UAVs (through onboard sensor suites)
- UAVs can communicate to each other or a ground control station without error or delay
- UAVs travel at a constant and known airspeed
- UAVs can maneuver without constraint on turn radius
- UAVs are not in hostile setting, i.e., no evasive or counter-detection maneuvers are required
- UAVs are not constrained by endurance limits on fuel/battery
- Goals (either strike targets or landing zones) are at known and stationary locations prior to commencing the path optimization
- There are no major terrain obstructions that would disrupt flight paths
- There are no environmental perturbations (e.g., wind) that affect flight paths

1.5 Contributions of this Thesis

1.5.1 Theoretical

The theoretical contributions of this thesis is the formulation of a network-based optimization model applicable for routing of a swarm of UAVs, combined with the application of agent-based models and other techniques to select and cut nodes in the network to decrease computational complexity. We explore several different methods for cutting nodes, includ-

ing an agent-based model that uses particle swarms to choose the most probable nodes that are relevant to the network formulation. Extensive numerical studies highlight the improvements and scalability achievable by use of these methods.

1.5.2 Operational

This thesis explores the trade off between centralized control and distributed control in terms of limited computational resources. Since a swarm can actually be a group of individual agents that can individually process given information and make decisions, are there benefits to decentralizing the path optimization process? This research is not restricted to aerial applications, but can also be applied to ground, surface, and subsurface swarms as well.

1.6 Organization of the Thesis

This chapter introduced the problem of optimizing the paths and sequence of UAV swarms for autonomous landing operations, with relevance to other UAV swarm coordination missions such as strike, and also included the motivation and relevant background. Chapter 2 describes the formulation of this problem as a network-based optimization model, and also presents the decentralized agent based model utilized for comparison. Chapter 3 develops a variety of novel algorithms for node and arc selection and selective restrictions on the network-based optimization model to extend the relevance of the solution space. Chapter 4 explores the agent based model and provides comparison of the network-based optimization model and agent based results. Lastly, Chapter 5 summarizes the presented work with recommendations for follow-on research.

CHAPTER 2:
MODEL FORMULATION

2.1 Scenario Description

Consider a situation, such as a saturation attack by an enemy force on a friendly asset, in which the United States responds by deploying a counter-swarm to defend against this swarm attack. Subsequent to the successful engagement, the friendly swarm of UAVs can either return to base or counter strike the enemy's positions. In both contexts, the UAVs reform into a "flock" and are poised to land in an efficient sequence or to prosecute assigned targets. As mentioned previously, the computational models for conducting either of these missions are similar, with the latter multi-target strike mission representing a relaxation of the sequenced landing and recovery operation. The model developed in this chapter highlights this more challenging formulation, graphically illustrated in Figure 2.1 with the collective UAV swarm preparing for recovery at three landing zones as an example.

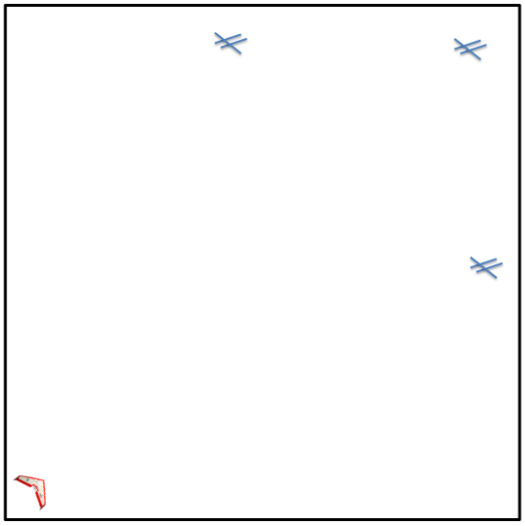


Figure 2.1: In this scenario, the landing zones are grouped together in relative close proximity (i.e., in the upper quadrant). This scenario serves as a baseline case for possible swarm encounters and thus provides the basis of much of the studies in this thesis.

2.2 Airspace Network Description

A network model describes a collection of nodes, denoted $n \in N$, joined together by edges, $e \in E$, which defines a graph, $\mathscr{G}(N,E)$. Additionally, information relevant to the model can be associated with nodes and edges, such as cost of traversing edges, capacity of flow along edges, supply/demand at nodes, etc. In the context of a minimum cost flow formulation , the optimization problem is solved by traveling through the graph, $\mathscr{G}$, and incurring cost

moving along edges from one or more source nodes to sink node(s). Min-cost flow models are specific examples of a network model in which the objective is to minimize this overall cost while maximizing the flow through the network.

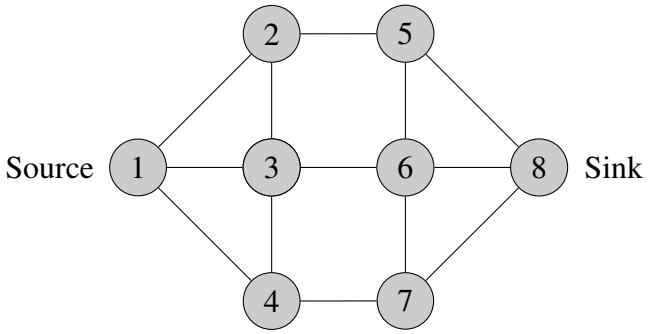


Figure 2.2: Illustrative example of a canonical network model (drawn using the `tikz` LaTeX package). In this instance we have an undirected graph, meaning that flow can occur in both directions along the edges. This is different from the directed graph in which flow can only occur in one direction.

As mentioned previously, we propose a network model with side constraints to describe the transit and landing (or targeting) of UAVs to airports (or targets). This model is similar to the min-cost flow model in that it seeks to minimize the objective function. Given the time-evolving nature of the problem, it is recognized that the underlying graph and corresponding data (e.g., capacities or availability of edges) may depend upon time. To address this temporal aspect, we construct a *time-expanded network model* [13]. Time expansion is the "copying" of all the nodes in the network, but associating each copy with a discrete time step from 0 to time horizon length, T. From henceforth, the proposed network model with side constraints detailed in this thesis will be referred to as the network model or network-based model synonymously.

The UAVs in the model start at a unique "UAV node" labeled $K_k,\ k = \{1,2,\ldots,w\}$ where w is the number of UAVs in the swarm. Each UAV node "holds" the supply of UAVs for distribution into the network. In this sense the UAV node is considered a "supply" node or "source node" in the standard terminology for network models. We choose not to use a single source node for our model (as is typically seen in this type of problem) primarily for bookkeeping reasons, as the post-processing of individual UAV paths is easier to extract if associated with unique sources for each UAV without changing the underlying network model. These UAV nodes are then connected to the UAVs' starting positions in the airspace, with arcs existing only for time step zero.

One UAV is allowed into a given airspace node at any given time step. This is assumed to provide sufficient UAV deconfliction of the airspace. From its origination point, the UAV is

assumed to be able to transit into any of its adjacent squares, as depicted in Figure 2.3. The UAV is assumed unable to remain in one place (e.g., fixed wing aircraft), and thus can not hover.

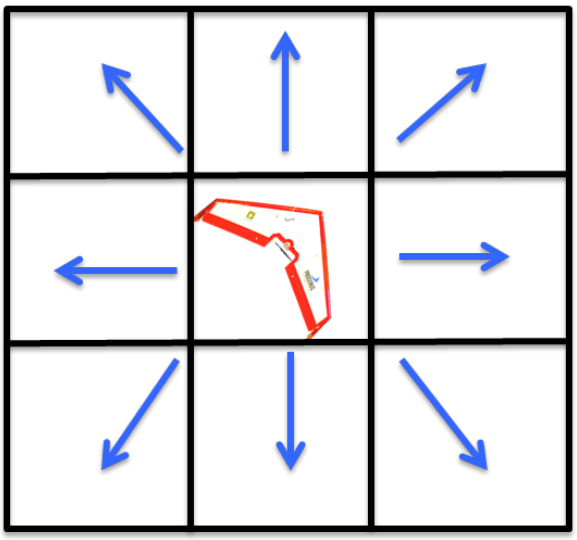


Figure 2.3: The UAV can travel to any of its eight adjacent nodes in the next time step.

Movement along an edge incurs unit cost and expends one time step to complete. The UAVs move through the network toward one or more airport nodes. The a^{th} airport node is denoted A_a, $a = \{1, 2, \ldots, p\}$ where p is the number of airports. The edge in the network extending from the airspace nodes to the airport node is similar to the UAV nodes, except that these arcs are replicated in time and thus appear in the time-expanded network. This allows UAVs to land at a given landing zone at any time step throughout the operation. Once a UAV reaches an airport node, it will travel to a special aggregating or sink node denoted $sink \in N$. The sink node contains all the demand for the system. This demand is represented by a negative number (in units of UAVs), which thereby forces the flow from the supply nodes (i.e., UAV nodes) to the *sink*.

The time expansion of this whole model dictates the duplication of edges (not nodes), such that the entire graph is duplicated for every time step with directed edges moving from nodes in one time step to those in the subsequent one.

In Figure 2.5 we display the concept of replicating the airspace network over time. With this representation the UAVs have to transit the space through time (in addition to the two-dimensional space) in order to reach an airport. This allows us to use well-documented network flow constraints by including a time index to the decision variables and edges. This augmented state is seen and presented in the formulation below.

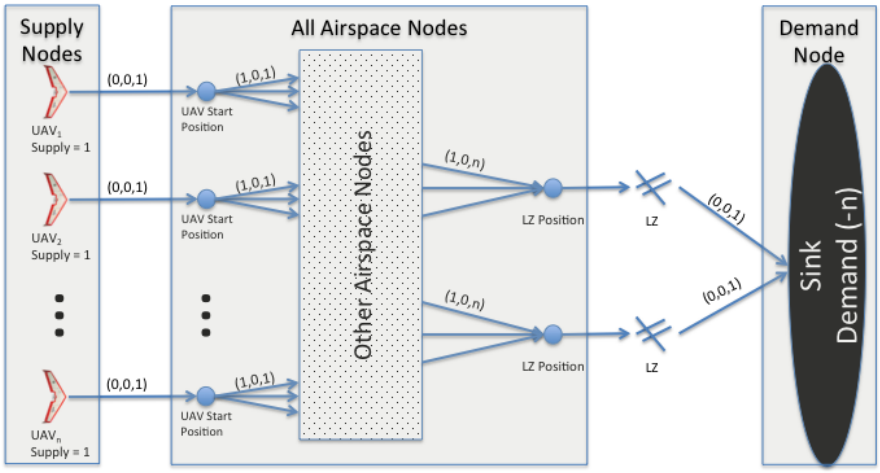


Figure 2.4: Representation of the network graph. The UAV nodes are represented as having a supply of one UAV per node. The edges possess associated data, which is represented by the tuple in the form of (cost, lower limit, upper limit).

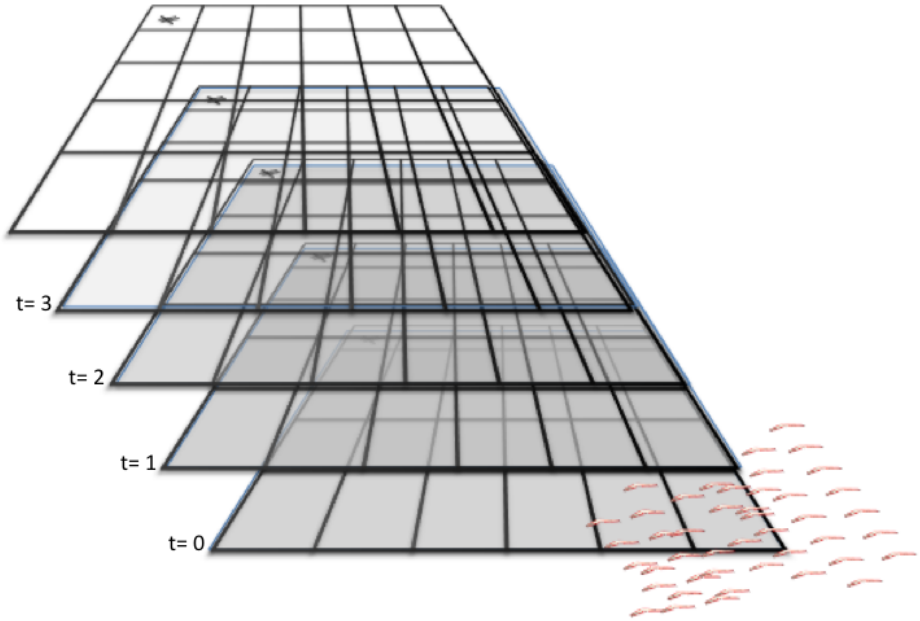


Figure 2.5: Illustration of the concept of replicating the airspace network over time. We can see that the UAVs enter the airspace at time zero in one big swarm, but that is the only place they are depicted since they are the supply traveling thought this network. The landing zone at the other side of the airspace, in contrast, is replicated in time, showing that the UAV can leave the airspace at any time at a landing zone.

2.3 Network-based Formulation

The following formulation is the mathematical representation of the model. It displays all the sets, indices, and data used to formulate the model using NPS format [21].

Sets

N : Set of nodes representing two-dimensional airspace

$K \subset N$: Subset of nodes corresponding to UAV locations within airspace

$A \subset N$: Subset of nodes corresponding to airport (or target) locations

E : Set of edges representing spatial and temporal connections between nodes

T : Indexed set of time steps

Indexes

$n, i, j \in N$: indices of nodes

$k \in K \subset N$: index of nodes containing UAVs

$a \in A \subset N$: index of nodes containing airport (or target) locations

$t \in T$: index of time step

Data

$arc_{i,j,t} \in E$: edge from node $i \in N$ to node $j \in N$ in time step $t \in T$

$Karcs_{k,n,t} \in E$: initial position of UAV $k \in K$ in time step $t = 0$

$Aarcs_{n,a,t} \in E$: position of airport $a \in A$ in time $t \in T$

$sink \in N$: sink node containing all demand [UAV]

$req_a \in A$: quantity of UAVs required to land at airport a [UAV]

$C_{i,j}$: cost to travel from node $i \in N \rightarrow j \in N$ [time]

UAV : constant for the total UAVs in swarm [UAV]

$\ell_{i,j,t}$: lower bound on $arc_{i,j,t}$ [UAV]

$\upsilon_{i,j,t}$: maximum capacity on $arc_{i,j,t}$ [UAV]

$\upsilon_{i,j,t} = 1$ if $i, j \neq sink$

$\upsilon_{i,j,t} = UAV$ if $(i, j) = (sink, sink)$ or $(a, sink)$

$b_{i,t}$: supply of demand at node $i \in N$ at time $t \in T$ [UAV]

$unsat_{i,t}$: unsatisfied demand at node $i \in N$ at time $t \in T$ [UAV]

G^a : capacity of airport $a \in A$ [time step]

Decision Variables

$$X_{i,j,t} : \text{ flow on } arc_{i,j,t} \text{ [UAV]}$$

Formulation

$$\min_X \sum_{arc_{i,j,t}} C_{i,j} X_{i,j,t} \tag{2.1}$$

$$\text{s.t.} \quad \sum_j X_{n,j,t} - \sum_i X_{i,n,t-1} = b_{n,t} \qquad \forall n \in N \tag{2.2}$$

$$\sum_{t=t'}^{t+G^a} X_{a,sink,t'} \leq 1 \qquad \forall t \in T,\ \forall a \in A \tag{2.3}$$

$$\sum_t X_{a,sink,t} \geq req_a \qquad \forall a \in A \tag{2.4}$$

$$0 \leq X_{i,j,t} \leq \upsilon_{i,j,t}, \text{ integer} \qquad \forall arc_{i,j,t} \tag{2.5}$$

Discussion

The objective function (Equation 2.1) expresses the total cost of all UAVs traveling though the system. Each constraint described in Equation 2.2 is a flow balance equation, ensuring any supply that enters a given node must exit the node with the exception of the UAV and sink nodes. Constraints 2.3 limit the number of UAVs that can land in a given landing zone $a \in A$ for a given number of time steps $t \in G^a$. Constraints 2.4 ensure that a prerequisite number of UAVs land at a given landing zone. Constraints 2.5 ensure that the upper bounds of a particular edge capacity are not exceeded. Constraint 2.5 implies that all $X_{i,j,t}$ are binary with the exception of the $(sink, sink, t)$ and $(a, sink, t)$ edges for airports $a \in A$. This is to allow UAVs to remain in the sink at all time steps without having to be removed. Note; that if Constraints 2.3 are removed, then this formulation represents a min-cost network flow problem describing the UAV simultaneous strike problem.

In this thesis we relaxed Constraints 2.5 to $0 \leq X_{i,j,t} \leq \upsilon_{i,j,t}\ \forall\ arc_{i,j,t}$, removing the integer requirement. This formulation does not guarantee an integer flow along any given arc; however, in all our empirical investigations, an integer flow was consistently achieved.

If a fractional flow were encountered along any arc then the relaxation would have to be removed and a mixed-integer program solution approach (e.g., branch and bound) would be applied.

2.4 Construction of the Network-based Model

The network-based model is created using a series of software tools[1]. First Matlab is utilized to create the nodes and edges. These nodes and edges are then passed to the following comma separated value (.csv) files.

- `NodesSet.csv` : The name of all the nodes in the system
- `EdgesSet.csv` : The name of all the edges represented as Node i , Node J, Time step
- `TimeSet.csv` : $T_0 \rightarrow T_t$
- `NodesData.csv` : Contains the supply and demand info for the UAV nodes and sink node
- `EdgeData.csv` : Contains all the edge data, i.e., cost, lower, and upper limits. Note this is not time step specific
- `UAVSet.csv` : The set of UAVs, as distinguished from the other nodes
- `APSet.csv` : The set of airports, as distinguished from other nodes in the airspace
- `APData.csv` : Contains information for the APSet, specifically the interval, and capacity information for ensuring that UAVs do not all land at once.

This information is then read into a software package called Generalized Algebraic Modeling Software (GAMS) [22]. GAMS then generates the model instance and passes it to CPLEX [23], a commercially available solver. The solution is then read back into GAMS and processed into another comma-separated value file. This flat file is imported into Matlab, from which data relevant to the analysis is processed and graphed, as illustrated in Figure 2.6.

[1] We learned and explored a variety of computation tools for use in this thesis, including Python and Java programming languages. Though Python has optimization-related packages and modeling languages such as Coopr and Pyomo, respectively, the lack of familiarity prevented significant use of Python. On the other hand, though Java provided an easier environment for coding the construction of the network, no suitable Java optimization libraries, including JaCoP, were found to be applicable to the problem sizes of interest. Ultimately we chose a combination of Matlab and GAMS.

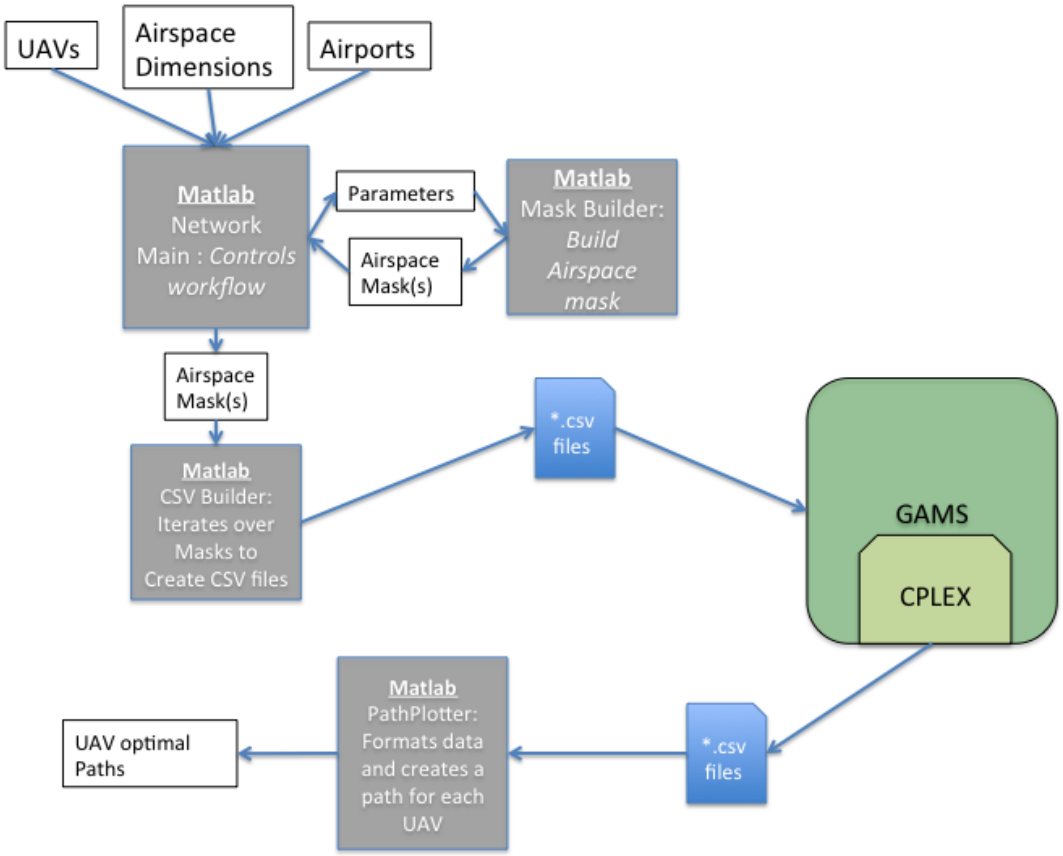


Figure 2.6: Diagram of the work flow for constructing the network, including the appropriate time expansion. We have written various Matlab scripts, which calls several function that help to construct the network. The model is then passed (in the form of a csv file) to GAMS and CPLEX for solving. The resulting solution in then returned to Matlab for analysis and visualization. All the code for this model can be accessed at: http://faculty.nps.edu/thchung (under software)

CHAPTER 3:
MODEL EXPLORATION

3.1 Introduction

The purpose of this chapter is to give a synopsis of the efforts taken to solve the network-based model. We quickly discover that there is an urgent need to reduce the state space by removal of any excess nodes and edges to make the problem more tractable. Without steps for node reduction, current computation limits prevent solving for airspaces in excess of 60×60 nodes. If we assume that each node represents an area that is 100 ft along its side, then 60 nodes approximately represents only one nautical mile. This is not operationally relevant considering that most military airfields reside in Class D airspace, which is nominally 5 nm in radius. All computations described in the following are performed on a custom-built computer[2] with the following specifications:

- 3.5 GHz Intel i7 CPU, over clocked to 4.5 GHz
- 32Gb of RAM
- 640GB SSD (solid state) harddrive
- 4TB HHD
- NVIDIA GeForce GTX 580 GPU with 3 GB RAM
- 3 TB of virtual memory

At the time of this thesis, these specifications reflect a very powerful, high-end workstation computer.

The remainder of this chapter is set forth as follows. First we explore a series of algorithms that are designed and used to select the nodes more efficiently that are inputs to the network-based model. Following that we compare the different algorithms according to a number of different metrics, including each algorithm's node reducing capability, objective function value, and computational memory. For reference, Figure 3.24 illustrates the resulting airspace networks for the different algorithms studied in this chapter.

3.2 Full Algorithm

The Full Algorithm describes the full model of the airspace where no nodes are removed. As mentioned we were only able to achieve results in a 60×60 patch of airspace. In Table 3.1

[2]Support for the purchase of this hardware and other thesis-related expenses was provided by the SSC-Pacific Student Research Fellowship, awarded to the author after a thorough external review and selection process.

we can see the approximate nodes and edges produced by certain airspace dimensions. Approximations of the nodes and edges can further be represented by the expressions below:

$$\text{Time} = \sqrt{X^2 + Y^2} + \left(Interval \times \left(\frac{\#\,UAVs}{\#\,Airports} \right) \right)$$

$$\text{Nodes} \approx X \times Y \times Time$$

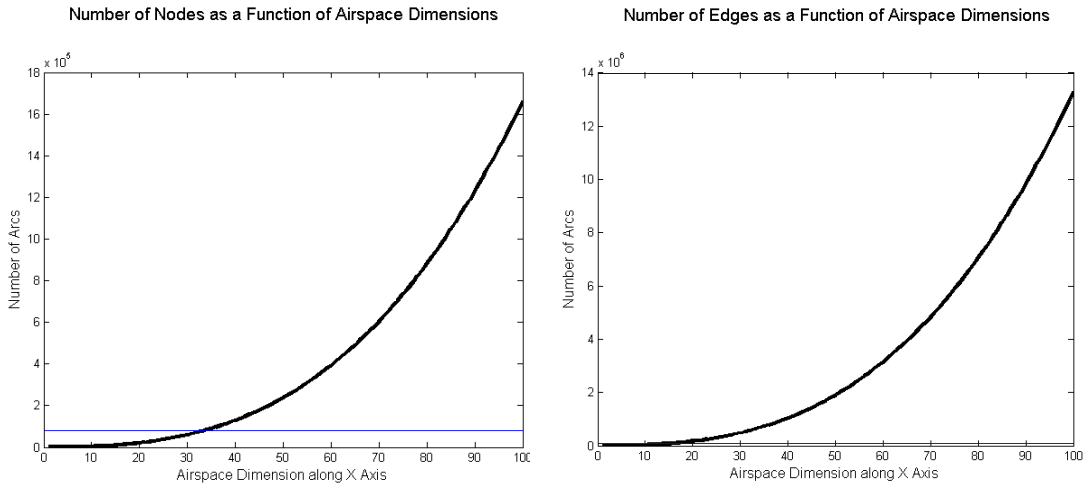
$$\text{Edges} \approx\ Nodes \times 8 \text{ (One edge for each direction)}$$

The following Table 3.1 and Figure 3.1 provide illustrative computation sizes as a function of different airspace dimensions using constant values of three landing zones, nine UAVs and a separation interval of two time steps.

Table 3.1: Number of Nodes and Edges created for various airspace dimensions. We can see that as the airspace grows the number of nodes and edges grow exponentially, quickly making the problem intractable.

Airspace Size	Nodes	Edges
30×30	6.50×10^4	4.80×10^5
90×90	1.26×10^6	1.01×10^7
180×180	9.039×10^6	7.23×10^7
300×300	4.041×10^7	3.23×10^8

The pseudocode described in Algorithm 3.1 is included for the reader's benefit to assist the reader's understanding of how we choose to set up the network before it is passed to GAMS. First, we iterate over the dimensions of the airspace, over every node in one time step. We then create a node, its node data, and edge data for each (X,Y) coordinate tuple. Then we iterate over all time steps until time T to create an edge from that node to all its adjacent nodes for each time step. This edge generation is possible even without explicitly creating the adjacent nodes, since we pass all node information as string data types into GAMS. Figure 3.24(a) also shows how the network includes the full airspace, on which any optimization problem requires significant computation resources.



(a) (b)

Figure 3.1: Graph of Nodes as airspace is increased. We can see from these graphs that there is a definite relationship between the the size of the airspace and the (a) number of nodes and (b) number of edges.

Algorithm 3.1 Generating full airspace network

X = maximum x component of the Airspace
Y = maximum y component of the Airspace
MaxTime = $||X,Y|| + \frac{2 \times \#UAV}{\#Airports} \times landingInterval$
for all $i = 1 \rightarrow$ X **do**
  for all $j = 1 \rightarrow$ Y **do**
    Create $Node_{i,j}$
    Create $EdgeData_{i,j}$
    for all t = startTime $\rightarrow$ endTime **do**
      if i+1 & j **then**
        Create $Edge_{Node_{i,j},Node_{i+1,j},t}$
      end if
      { Use an if statement to create an edge in each direction}
      {(i+1,j),(1+1.j+1),(i,j+1),(i-1,j+1),(i-1,j),(i-1,j-1),(i,j-1),(i+1,j-1)}
    end for
  end for
end for

Figure 3.2 illustrates the UAV paths through the airspace in respect to time. The x and y axes represent the physical airspace, whereas the z axis represents time steps. The UAV paths are represented by the different color lines and points. There is a red vertical line that represents

the (stationary) landing zone locations. For this illustrative example, the northeast corner of the airspace is considered the "sink" for all UAVs. We can see from the figure that all the UAVs flow into the landing zones, and then from there into the sink, remaining in the sink thereafter. We choose to use a 30×30 representation for this example for simplicity, although more expansive and relevant airspace dimensions will be covered in the sequel.

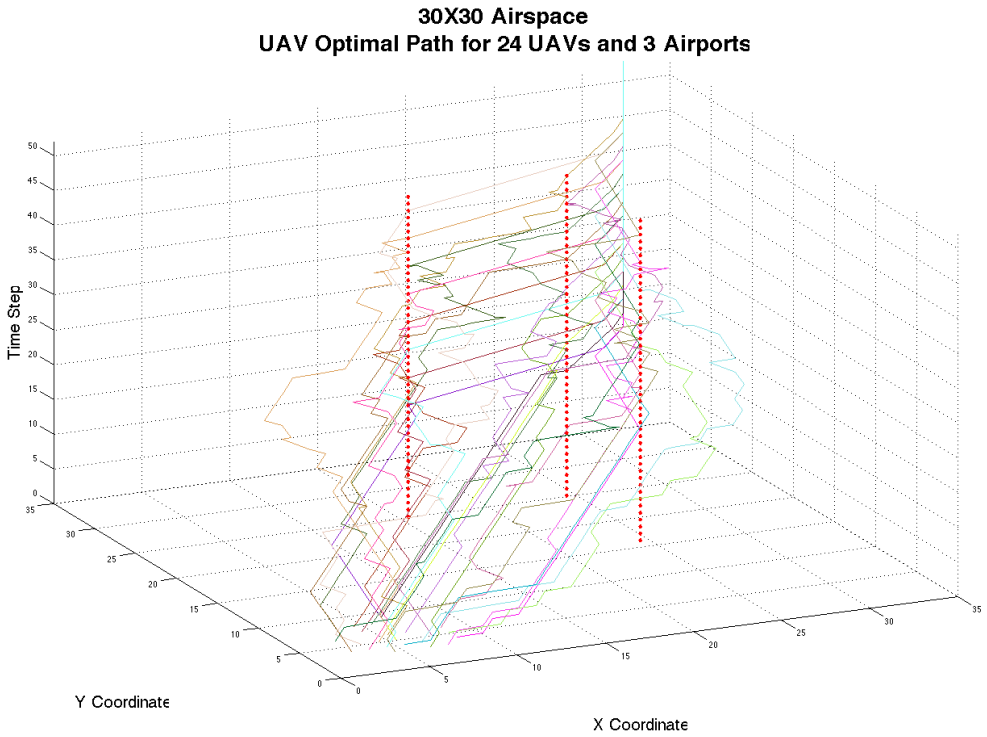


Figure 3.2: Optimal Path of UAVs in 30×30 Airspace using Full Algorithm

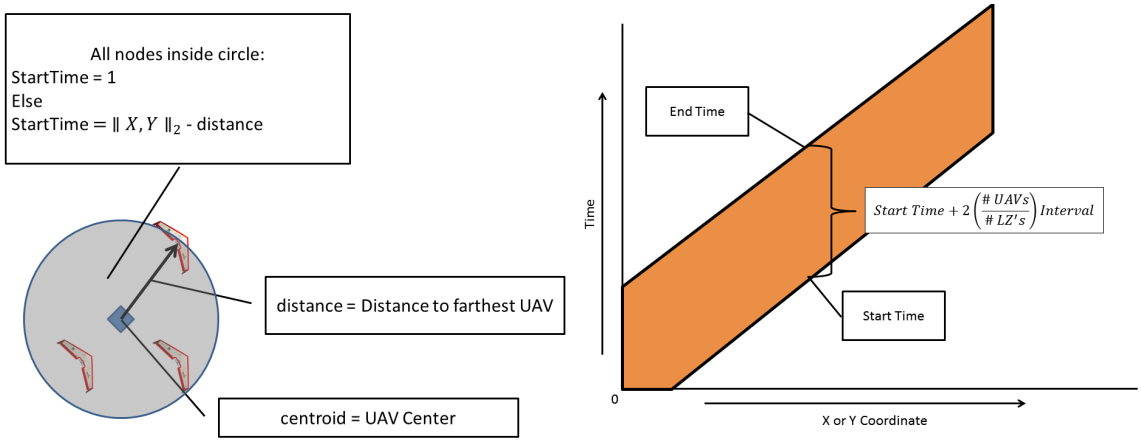
3.3 Time Slice

Our first attempt to reduce the number of nodes to make the problem more tractable is to eliminate nodes based on when they could be encountered by any UAV, i.e., the reachable set as a function of time. Logically there is no reason to have a node if there is no UAV that could possibly enter that node given the amount of time passed. In Figure 3.1 we can see that we eliminate a significant number of nodes in a 60×60 airspace by using this reachable node set notion, reducing from 356430 to 97230 nodes. In Figure 3.4, we can see the optimal path of each UAV displayed over time in this reduced airspace.

In this approach we iterate over the (X,Y) pairing as done previously. The key difference is as we iterate over the time steps we change the starting time step and ending time step. For this we do simple vector math to determine when a UAV could possibly encounter a node. To perform this calculation, first we determine the center of the UAV swarm. We then

determine the distance from that centroid location to the farthest UAV in the swarm. This creates a circle around the UAV start positions, such as in Figure 3.3. All points in this circle receive a start time of $t = 1$. This area is where all the UAV start positions reside and where they must be at time step one. We then start to restrict the start time based on the Euclidian distance from the edge of the starting circle to the node.

We then need to give the UAVs enough time to navigate away from the node to the landing zone. Realizing that multiple UAVs could transit over this node at different times we need to provide a sufficient time window for this to happen. This is accomplished by adding to the start time an appropriate length of time, computed as follows: the number of UAVs that are headed to a particular airport (e.g., in the uniform allocation case, $\frac{\#\text{ UAVs}}{\#\text{ Airports}}$, multiplied by the specified time interval, because it is reasonable to expect the UAVs to hold or posture to meet their interval and landing times. This assumption accounts for the time needed to safely transit. A factor of two is included to provide a buffer, representing an increase in the number of choices of nodes for each UAV throughout the duration of the transits. Modifications to the previous pseudocode are shown in Algorithm 3.2, reflecting the inclusion of nodes to the network only if reachable by any UAV within a given time window.



(a) Start Time (b) End Time

Figure 3.3: Start time and End time determinations for time slice approach. In (a) we see a top down representation of how we determine the first time step, where each node inside the the circle is to be included in time step $t = 1$. In (b) we see how the time slice cuts nodes along one axis direction in relation to time.

Algorithm 3.2 Generating time-sliced airspace network

Centroid = Center location of UAV swarm
distance = Distance from farthest UAV from center of Swarm
ij = (i,j) vector
X = maximum x component of the Airspace
Y = maximum y component of the Airspace
MaxTime = $||X,Y|| + \frac{1.5 \times \#UAV}{\#Airports} \times landingInterval$
{ this is assuming a worst case example}
for all $i = 1 \rightarrow$ X **do**
  for all $j = 1 \rightarrow$ Y **do**
    Create $Node_{i,j}$
    Create $EdgeData_{i,j}$
    if $||ij - Centroid|| \leq distance$ **then**
      startTime = 1
    else
      startTime = $||ij - Centroid|| - distance$
    end if
    endTime = min$(MaxTime, ijNorm + \frac{2 \times \#UAV}{\#Airports} \times landingInterval$
    for all t = startTime $\rightarrow$ endTime **do**
      if i+1 & j **then**
        Create $Edge_{Node_{i,j},Node_{i+1,j},t}$
      end if
      { Use an if statement to create an edge in each direction}
      {(i+1,j),(1+1.j+1),(i,j+1),(i-1,j+1),(i-1,j),(i-1,j-1),(i,j-1),(i+1,j-1)}
    end for
  end for
end for

In Figure 3.4 and 3.2, we can see that the UAVs are using a large portion of the airspace initially at the early stages of their transits to conservatively ensure their spacing and interval, well before they approach the landing zone. In some cases the last UAV to land executes several sharp "S" turns to prepare itself for landing in sequence.

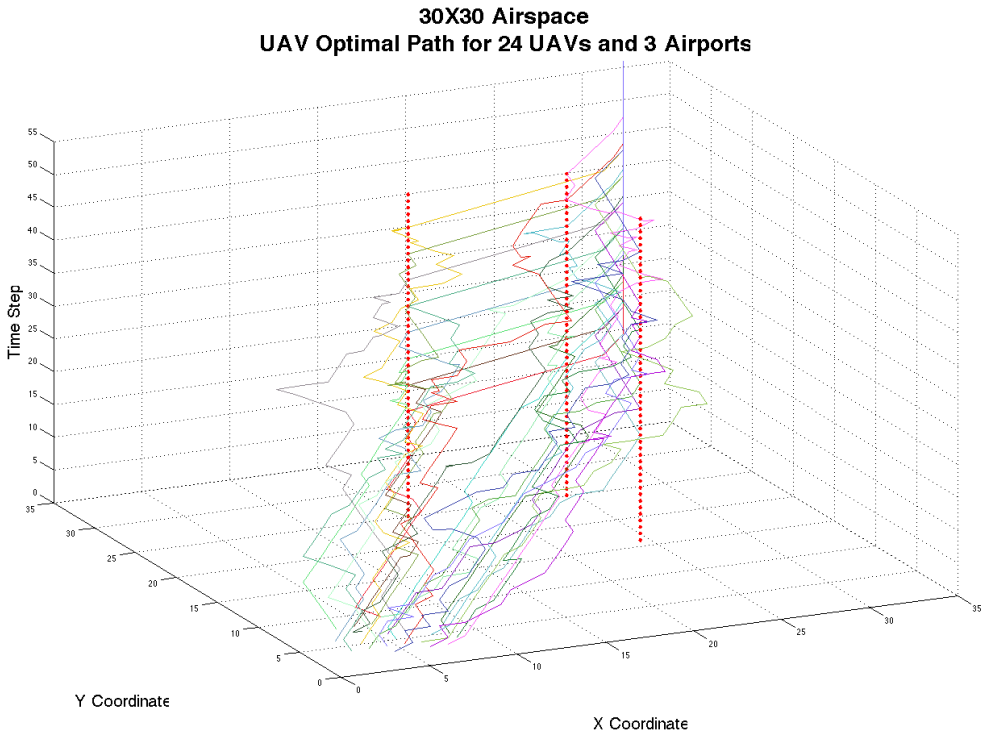
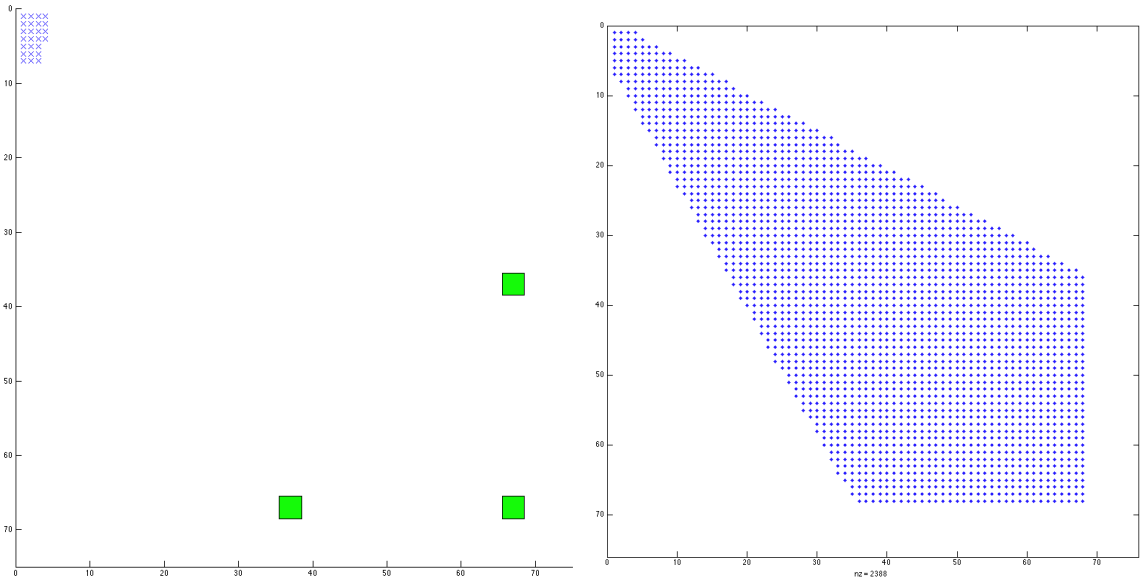


Figure 3.4: Optimal Path of UAVs in 30× 30 Airspace using Time-sliced algorithm. We can see that the UAVs start to spread out quickly as they set up for landing at the appropriate interval.

3.4 Convex Hull

In an effort to continue to reduce the number of nodes, we now present the use of the convex hull of the airport locations and the starting positions of the UAVs. From [24, 25], a convex hull is a polygon in which all points are either inside or on the line of the polygon. If a line segment is drawn between any two points in this polygon, then all points on that line segment must lie within the polygon as well. The function used to determine a convex hull of the points has a computational complexity of $\mathcal{O}(n \log n)$ [25]. Using this definition we place a square area around the landing zone to define a local region of responsibility for each airport, and created a convex hull from the outermost points of all the UAVs' starting locations and the landing zone square regions. This creates an airspace representation that, for our baseline scenario, looks like Figure 3.5. This technique does have its limitations, however; for example, if the landing zones were located at the far corners of the airspace, then we gain little, since we lose the advantage of eliminating all the edges from either side of the swath seen in the baseline scenario. This limitation highlights the fact that such network reduction approaches will likely depend highly on the specifics of the scenario (the effects of which are investigated later in this chapter). After the creation of the convex

hull, the resulting airspace is stored in a data structure we term a *mask*, which is a simple boolean matrix of ones and zeros. Figure 3.6 shows a simple example of a mask for a 10×10 airspace. A 1 entry indicates that the node is to be included in the network model, 0 indicates otherwise.



(a) UAV and airport locations

(b) Resultant convex hull

Figure 3.5: A graphical depiction of how the convex hull of the airspace is created. (a) positions of UAVs denoted by and "x" and landing zones depicted as squares. (b) The resulting polygon producing the convex hull of the airspace.

$$\begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Figure 3.6: An example of a mask. 1 indicates that the node is included in the computation, a 0 indicates that the node is not included.

The significant difference between this technique and that described by Algorithm 3.2 is that

we are cutting nodes on either side of airspace. In this instance we add a logical check to see if a particular node is in the convex hull. This check is accomplished by querying the resultant mask created by the convex hull. If that (X,Y) pair in the mask is a 1, we include it in the node list and create edges in the same manner discussed previously, as outlined in Algorithm 3.3 and depicted in Figure 3.24(c).

Algorithm 3.3 Generating the airspace network using the convex hull approach

vertices = blank Array
for all $a = 1 \rightarrow \#Airports$ **do**
  for all $i = 1 \rightarrow 4$ **do**
    vertices = Array; Corners of Airport Polygon
  end for;
end for;
vertices = vertices + UAV Positions
CH = convex hull of (vertices)
{run earlier time slice algorithm but add a logical check } {to determine if the node and its adjacencies are infact in the convex hull}
for all X and Y pairs **do**
  if $(i, j) \in CH$ **then**
    Create nodes
    if $(i+1 \ \& \ j) \in CH$ **then**
      Create $Edge_{Node_{i,j},Node_{i+1,j},t}$
    end if
    { Use an if statement to create an edge in each direction}
    {(i+1,j),(1+1.j+1),(i,j+1),(i-1,j+1),(i-1,j),(i-1,j-1),(i,j-1),(i+1,j-1)}
  end if
end for

Figure 3.7 shows the resultant path of the UAV to the convex hull algorithm. We can see that the UAVs still are using the initial portion of the airspace to gain position and interval; however, the paths comprise a number of smaller "S" turns (i.e., back and forth holding maneuvers) until the airspace starts to open up, after which the UAVs start to make larger turns and timing corrections. This is the result of the decrease in node availability on the sides of the airspace that have been removed before the network is sent to the optimization software.

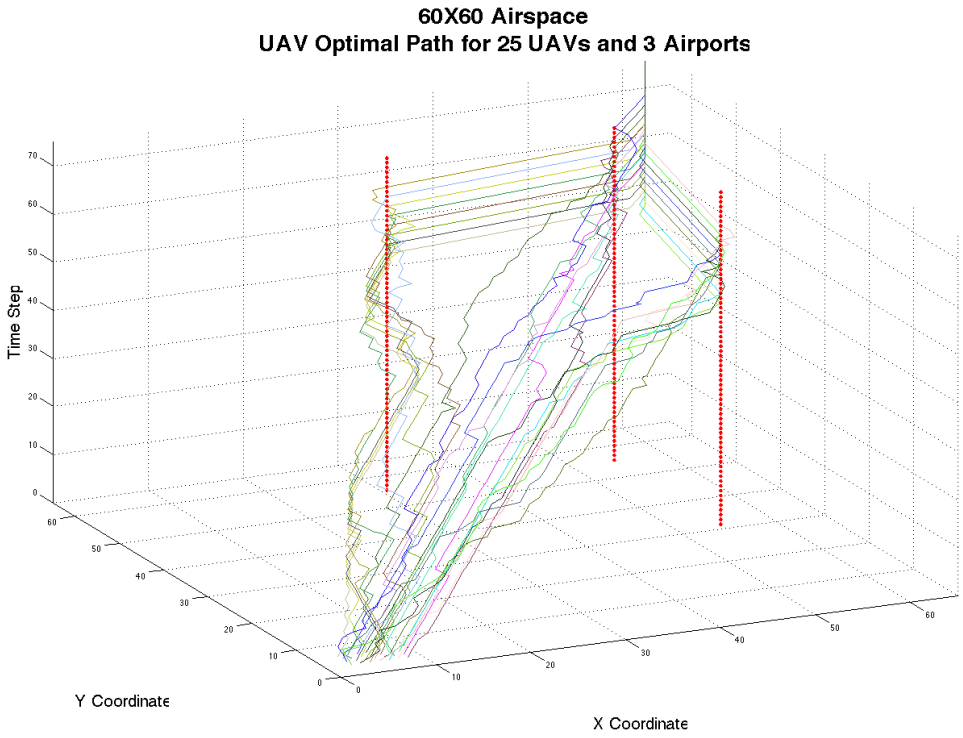


Figure 3.7: Optimal path of UAVs in 60×60 airspace using Convex Hull. The UAVs start to spread out early in the airspace at they position for landing

3.5 Multiple Convex Hulls

This technique is the similar to the convex hull construction discussed previously (c.f. Algorithm 3.3) but in this multiple convex hull approach, such a convex hull is constructed for each airport individually. The union of the resulting sets of nodes is then taken to represent the airspace network (see Figure 3.24(d)). We then utilize the time slicing algorithm to further cut nodes. This approach provides a significant improvement on the previous attempts since it is able to remove the nodes between the airways to the airports as well at on the sides, and is summarized in Algorithm 3.4.

Figure 3.8 depicts the resultant paths produced from the multiple convex hull algorithm. We see that the paths utilize the entire airspace. The UAVs start to maneuver to achieve their desired intervals and positions very early, again using those small "S" turns. Of interest is that there are surprisingly few 180° turns. These paths actually follow a realistic flight dynamic, even though such constraints on the dynamics are not explicitly included in this optimization model. Future studies may examine the impact of incorporating detailed flight dynamics, such as Dubin's vehicle models or other bounded motion models.

Algorithm 3.4 Generating airspace network using multiple convex hulls approach

for all $a = 1 \rightarrow \#Airports$ **do**
  vertices = Blank Array
  for all $i = 1 \rightarrow 4$ **do**
    vertices = Array; Corners of Airport Polygon
    vertices = vertices + UAV Positions
    AirportCH = convex hull of (vertices)
    CH = $CH \cup AirportCH$
  end for;
end for; {run earlier time slice algorithm but add}
if $(i, j) \in CH$ **then**
  Create nodes and edges
end if

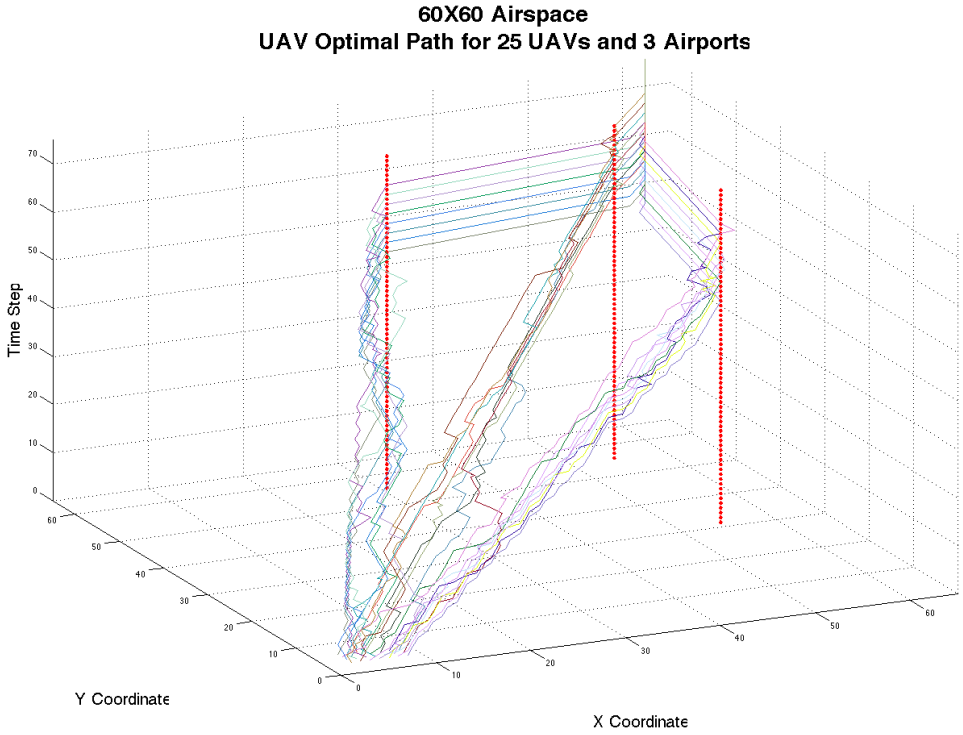


Figure 3.8: Optimal path of UAVs in 60×60 airspace using Multiple Convex Hulls. We can see the the path deviations and correction are small. This is because the UAVs are starting to work into an interval early while they still have enough space to maneuver.

3.6 Agent-Based Masks Algorithm

In the agent-based mask presented in this section, we use a combination of agent-based simulation and the convex hull technique discussed previously. In this instance we randomly initialize a large number of agents in the vicinity of the starting location of the UAVs. We

then randomly assign each agent a landing zone to "fly" to. The agents are given simple rules that they follow to move toward their assigned landing zones with some random noise to provide dispersion and a wider path. Over the course of Monte Carlo simulation runs, we then track the visited airspace that the agents occupied in each time step and created a convex hull for that occupied airspace. This procedure gives us natural time slicing throughout the network, while still providing enough space for the UAVs to move about without significant deconfliction issues. Algorithm 3.5 describes the methodology for constructing the network using this agent-based approach, and Figure 3.24(e) graphically illustrates the resulting network.

Algorithm 3.5 Generating airspace network using Agent-Based Masks approach

Time = theoretical Maximum Time
Agents = Array, UAV locations + randomly generated Agents around UAV start Position
Assignments = randomly assign each agent a landing zone
for all t ¡= Time **do**
  for all $a = 1 \rightarrow \#Agents$ **do**
    goalIJ = $Agents_a$ assigned landing zone location
    continue = true
    Coeff = log10(T)/4 {controls variance on random noise created around movement vector, as Agents get closer to assigned landing zone variance increases }
    drive = $goalIJ - Agents_a$ { Agents Movement vector toward the assigned Airport}
    if drive == 0 **then**
      drive = (0,0) { this is so we don't divide by 0}
    else
      drive = $||drive||$
    end if
    $Agents_a$ = $Round(Agents_a + drive + Coeff * RandomNormal(0,1))$
    {Standard Normal distribution provides noise for agent movement}
  end for
  CH = convhull(Agents)
  Masks(T).map = MASK(CH)
  Masks(1).map = $Masks(1).map \cup Masks(T).map$
end for

Figure 3.9 depicts the resultant paths produced from the agent-based masks algorithm. This plot is very similar to the convex hull plot, as the UAVs use much of the airspace initially, with frequent but small course corrections, but spread out more as more airspace becomes available.

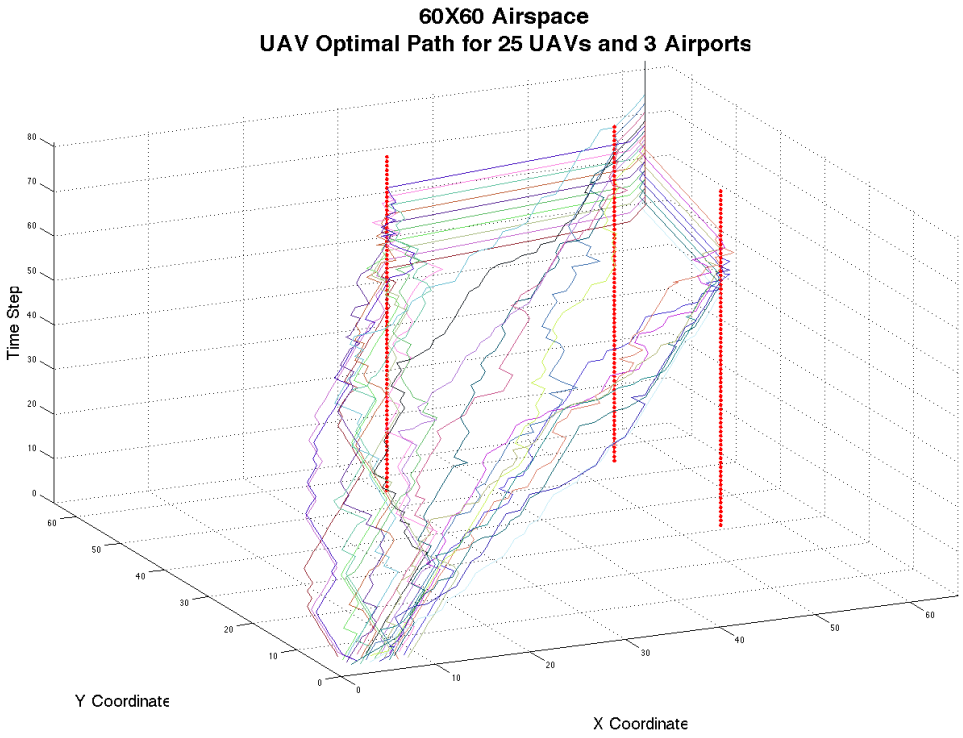


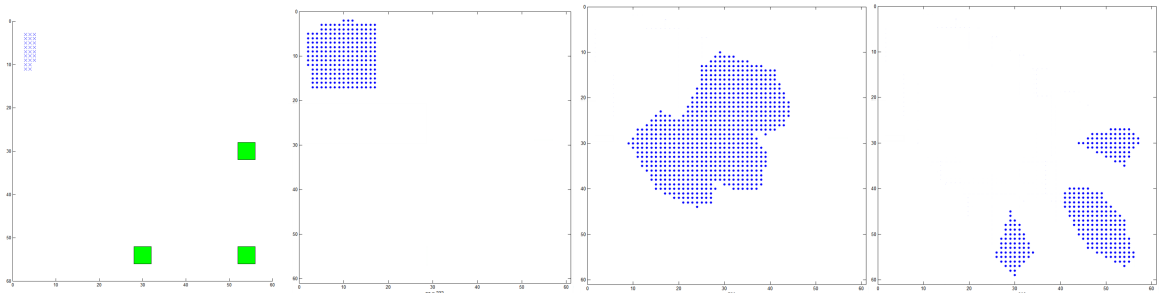
Figure 3.9: Optimal path of UAVs in 60×60 airspace using the agent-based masks approach.

3.7 Organic Masks

This approach is a culmination of our previous explorations. In this organic mask algorithm we use the same agent-based model that was developed for the mask algorithm above, combined with the multiple convex hull algorithm, by keeping track of the individual groups of UAVs that are traveling to their assigned landing zone. In Figure 3.24(f) we can see that the airspace qualitatively appears similar to the result of the multiple convex hull approach, looking like a claw or rake. In this method, we also remove a considerable number of nodes, by encapsulating viable routes (while eliminating unlikely ones) that the UAVs follow as they travel along the network.

The organic mask effectively "grows" the network, based on agent movements flowing through the network. In this approach we again assign the agents a landing zone as done in the previously described agent-based masks approach. As we progress in time we apply a movement vector which forces the agents to move toward their respective landing zones. We then add random normal noise to this vector. All the agents traveling to a particular landing zone are considered a separate set or flock of UAVs, such that for each time step, we construct the convex hull of each of these sets and create our mask. Figure 3.10 depicts the progression of the mask construction through time, with the pseudocode described in

Algorithm 3.6.



(a) $t = 0$ (b) $t = 1$ (c) $t = 20$ (d) $t = 40$

Figure 3.10: Organic Masks algorithm: convex hull of agent sets as they progress though the airspace. Each frame represents an individual mask in a particular time step. We can see how the nodes are selected at the agents move through the airspace.

Algorithm 3.6 Generating airspace network using organic masks approach

```
Time = theoretical Maximum Time
Agents = Array, UAV locations + randomly generated Agents around UAV start Position
Mask(T).map = empty mask
Assignments = randomly assign each agent a landing zone
for all t ¡= Time do
  for all a = 1 → #Agents do
    ******Move all agents as in Masks Algorithm*****
  end for
  for all  k = 1:size(Airports) do
    index = set of all agents assigned to Airport_k
    CH = convexhull(Agents_index)
    Masks(T).map = Masks(T).map ∪ MASK(CH)
    Masks(1).map = Masks(1).map ∪ Masks(T).map
  end for
end for
```

Figure 3.11 depicts the resultant paths produced from the organic masks algorithm. Here we see very little "room" initially for the UAVs to maneuver and position themselves early in the evolution, which results in some congestion later on akin to traditional holding patterns as the UAVs loiter around the airports waiting for their respective turns to land. Also in this instance, we begin to see flight dynamics that are less feasible in actual UAV platforms, exhibiting significantly sharp turns, where in previous examples, the paths appear to be more dynamically feasible.

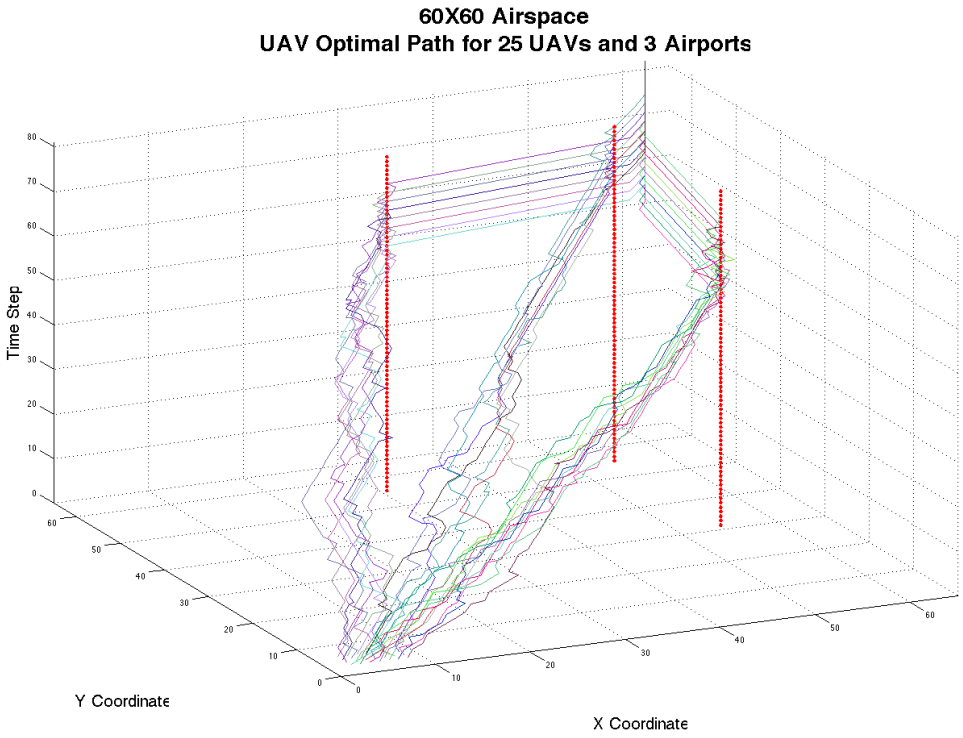


Figure 3.11: Optimal path of UAVs in 60×60 airspace using Organic Masks

Figure 3.24 graphically shows the airspace network for the full airspace model, alongside the five proposed network reduction approaches in this thesis. Nodes included in the airspace network is indicated by the individual black dots, all for a 60×60 airspace. One can readily see the significant differences between the various networks, with fewer nodes leading to solutions for more operationally relevant airspace contexts.

3.8 Algorithm Comparison

In this section we compare the different models as they pertain to five different scenarios, that is, five different spatial configurations of the landing zones and their numbers. Each of these scenarios highlights the different models' respective strengths and weaknesses, while allowing us to identify which algorithm performs best overall. This investigation also allows us to realize the computational trade offs of each algorithm, enabling us to make a better choice for a given class of scenarios. The solution to the network-based model using the full algorithm (i.e., the full airspace) is taken to be the true optimal solution, since it contains all of the nodes and edges that can possibly be included. We compare all algorithms against this benchmark optimal solution.

3.8.1 Scenario Constants

The following parameters are held constant for each of the scenarios to ensure that the focus of the study and resulting analysis remain on the impact of the configuration of the landing zones.

- Number of UAVs : 25 UAVs
- Size of airspace : 60×60 cells
- Landing interval: 1 time step

The following factors vary per scenario, but remain constant in the application of each algorithm:

- UAV start locations: Each UAV will be assigned a specific start location
- Landing zone locations: Each landing zone is assigned a specific location
- Required # of UAVs: The number of UAV required to land at each airport

For each scenario, the presented results tables provides a comparison for the different network construction algorithms. Included are the number of nodes and edges for the constructed airspace network to provide a measure of the state space. Computational measures of the amount of memory (in megabytes) and runtime in seconds[3], and mission performance measures of the objective function value (Equation 2.1) and the Time Out of System (TOS) or the time the last UAV lands (in time steps since the start of the mission) are also shown.

3.8.2 Scenario One: Group of Landing Zones

Scenario One is what we have explored thus far in the previous sections as the baseline scenario. In this instance we have a small group of three landing zones in relatively close proximity (i.e., they reside in the same quadrant). This configuration is meant to accentuate the effect of the convex hull and agent-based masks algorithms. Since the landing zones are close together we see significant node cutting using these two techniques.

[3]Runtime is the total execution time of the problem from generating the airspace in Matlab, GAMS reading the .csv files and receiving a solon from CPLEX. Solve time is strictly the time required for CPLEX to solve the network-based model, once it has been properly instantiated and passed to CPLEX. Example of solve time compared to runtime from Scenario One include:
Full: solve = 1429 seconds ; runtime = 5282 seconds
Convex Hull: solve = 19 seconds ; runtime= 158 seconds
Multi Convex Hull: solve = 8 seconds ; runtime = 64 seconds
We can see that solve time is significantly less than the runtime in all cases. The runtime could be decreased by interfacing directly with the solver, using data structures that stay resident in the generating language, and/or using a compiled language.

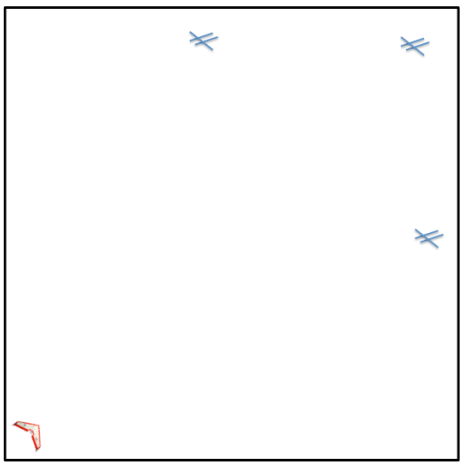


Figure 3.12: Graphical Depiction of Scenario One. UAVs are located at one corner of the airspace, and the landing zones are located in close proximity to each other in the other corner of the airspace. The scenario involves 25 UAVs, three landing zones, and at least eight UAVs required to arrive at each landing zone.

Table 3.2: Results of different algorithms for Scenario One

Algorithm	**Nodes**	**Edges**	**Memory**	**Obj Value**	**TOS**	**Runtime (sec)**
Full	356430	2664413	83938	1381	65	5282
Time Slice	96931	725253	83815	1554	81	4816
Convex Hull	42366	326537	16563	1430	70	158
Multi Convex Hull	26225	164286	5213	1430	70	64
Agent-based Mask	40437	296496	26821	1449	70	192
Organic Mask	21982	141799	18448	1442	69	110

3.8.3 Scenario Two: Dispersed Landing Zones

Scenario Two has three landing zones similar to Scenario One; however, the landing zones are now placed in the three corners opposite the UAVs' starting position. The takeaway from the simulation studies of this scenario is to mitigate the node reduction advantages of the the convex hull and mask algorithms, while highlighting the effectiveness of both the multiple convex hull and organic mask algorithms.

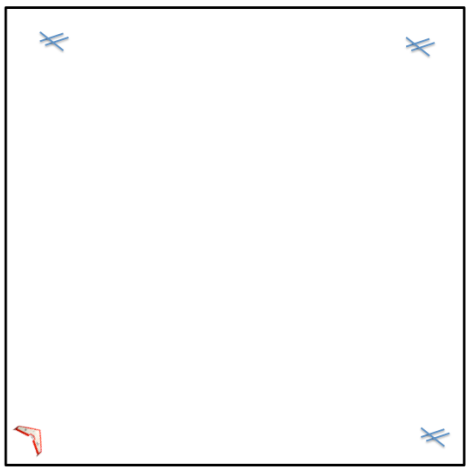


Figure 3.13: Graphical Depiction of Scenario Two. UAVs are located at one corner of the airspace, and three landing zones are located in each of the other far corners, which highlights the benefits of multiple convex hull and organic masks approaches over their respective counterparts. This scenario also involves 25 UAVs, three landing zones, and at least eight UAVs required to arrive at each landing zone.

Table 3.3: Results of different algorithms for Scenario Two

Algorithm	**Nodes**	**Edges**	**Memory**	**Obj Value**	**TOS**	**Runtime (sec)**
Full	2665008	353194	83953	1381	66	5246
Time Slice	106420	725848	83837	1501	80	4543
Convex Hull	73924	574442	48931	1501	80	1875
Multi Convex Hull	21733	147572	4496	1501	80	63
Agent-based Mask	173887	1322684	62776	1455	69	3058
Organic Mask	47414	320813	20064	1437	69	123

3.8.4 Scenario Three: Overflight of Landing Zone

This scenario now has four landing zones, but one of the landing zones is directly in the path of another. In other words, two UAV groups for these coinciding paths now must transit and deconflict in the same restricted airspace for the two different airports. This feature is to observe the impact of a more challenging deconfliction setting on the different algorithms, and see if there is an effect on the ability to find an optimal solution.

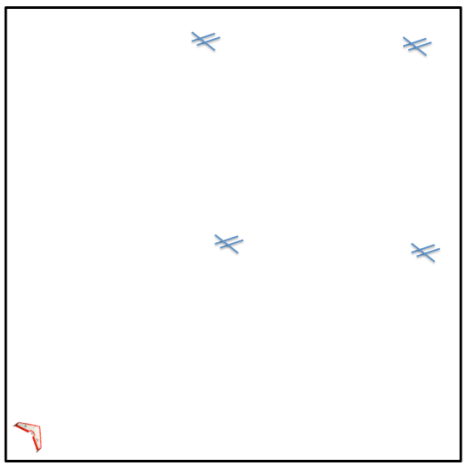


Figure 3.14: Graphical Depiction of Scenario Three. UAVs are located at one corner of the airspace, and the landing zones are located in close proximity to each other in the other corner of the airspace. The fourth landing zone is placed in line with the farthest landing zone, requiring overflight of transiting UAVs. This scenario also involves 25 UAVs, but four landing zones, and at least six UAVs required to arrive at each landing zone.

Table 3.4: Results of different algorithms for Scenario Three

Algorithm	**Nodes**	**Edges**	**Memory**	**Obj Value**	**TOS**	**Runtime (sec)**
Full	342477	2584443	81462	1178	62	6420
Time Slice	92182	618291	81343	1339	76	1187
Convex Hull	41304	278721	16079	1339	77	162
Multi Convex Hull	24003	146287	5492	1339	77	67
Agent-based Mask	56746	426880	272590	1127	62	241
Organic Mask	47414	320813	20064	1437	69	105

3.8.5 Scenario Four: Airports in Each Corner

This scenario has four landing zones, such that the UAVs are initially located towards the center of the airspace and the landing zones are located at each distant corner of the airspace. The purpose of this scenario is two fold. First, this experiment allows all of the airports to be equidistant from the stating position of the UAVs and from each other, which maximizes their spatial dispersion and may lead to an improvement in performance due to earlier segregation of the UAVs. Second, the UAVs are not starting in the corner and this modification allows us to explore the flexibility of the algorithms, having parametrized not only landing zone locations but also starting locations for UAVs.

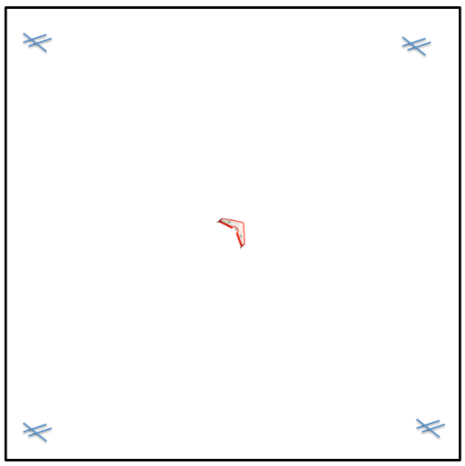


Figure 3.15: Graphical Depiction of Scenario Four. UAVs are located at center of the airspace, and the landing zones are located in each of the four corners of the airspace. This allows airspace to be equidistant from the center of the UAVs, and tests algorithms for their effectiveness in dispersing the swarm in separate directions. This scenario also involves 25 UAVs, but four landing zones, and at least six UAVs required to arrive at each landing zone.

Table 3.5: Results of different algorithms for Scenario Four

Algorithm	**Nodes**	**Edges**	**Memory**	**Obj Value**	**TOS**	**Runtime (sec)**
Full	342477	2584443	81462	693	36	4782
Time Slice	55734	350747	81328	756	40	4233
Convex Hull	40665	263491	42722	756	40	1056
Multi Convex Hull	14815	102098	3338	756	39	54
Agent-based Mask	214982	1648721	59745	761	39	3124
Organic Mask	17510	108400	10410	750	37	78

3.8.6 Scenario Five: Five Airports at Multiple Depths

This scenario introduces five landing zones that are at different depths and proximity. This more challenging setting explores the node selection capability of the algorithms with many landing zones at a variety of distances. The scenario also tests the network-based model's ability to assign UAVs to five landing zones. As such, this scenario provides the most difficulty from both the set-up and the computational solution perspectives.

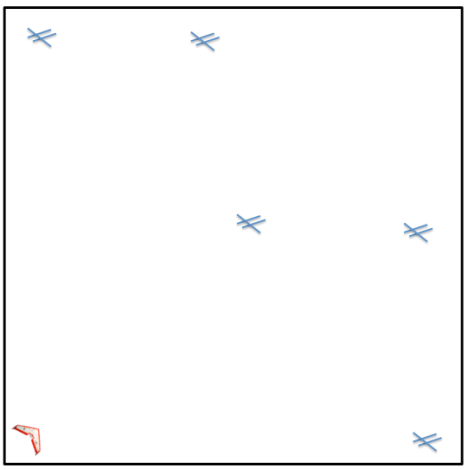


Figure 3.16: Graphical Depiction of Scenario Five. UAVs are located at one corner of the airspace, and the landing zones dispersed throughout the airspace in both close proximity to each other, but also spread to cover most of the airspace. This scenario also involves 25 UAVs, five landing zones, and at least five UAVs required to arrive at each landing zone.

Table 3.6: Results of different algorithms for Scenario Five

Algorithm	**Nodes**	**Edges**	**Memory**	**Obj Value**	**TOS**	**Runtime (sec)**
Full	335360	2530789	79779	1115	54	5206
Time Slice	85108	564421	79677	1175	50	4636
Convex Hull	52385	356214	29930	1175	50	256
Multi Convex Hull	24029	143148	6353	1173	59	69
Agent-based Mask	130528	986367	37529	1175	54	740
Organic Mask	48654	327009	22594	1074	56	136

3.8.7 Discussion

There are trade offs between the different algorithms. The Full algorithm has the advantage of giving us the true optimal solution with no restrictions on node or edge choice to gain the optimal path. However; the problem is too large for efficient computation as the airspace and the number of UAVs increases. The multiple convex hull and organic mask algorithms have the most node cutting power, and are able to reduce the computation enough to make larger problems more tractable (see Figure 3.18).

This computational savings, however, comes at the expense of ensuring optimality of the solution (with respect to the full airspace network model). For example, in the case of the multiple convex hull algorithm, the resulting measure for the total time in system can be

approximately 30% greater than the Full solution, as depicted in Figure 3.22. The organic mask, though having a solution and objective value that is closer to the Full benchmark values, is more computationally expensive than the multiple convex hull approach by an average of 10%, and in most cases violates the "required" constraint (c.f. Equation 2.4), allowing fewer UAVs to land at any given landing zone than is required (see Figure 3.17). This requirement constraint is consistently broken in the various test scenarios explored. This violation appears to occur because the nodes that are cut do not offer enough "maneuver room" for the UAVs; the closest landing zone takes more UAVs than the other landing zones to alleviate this space restriction.

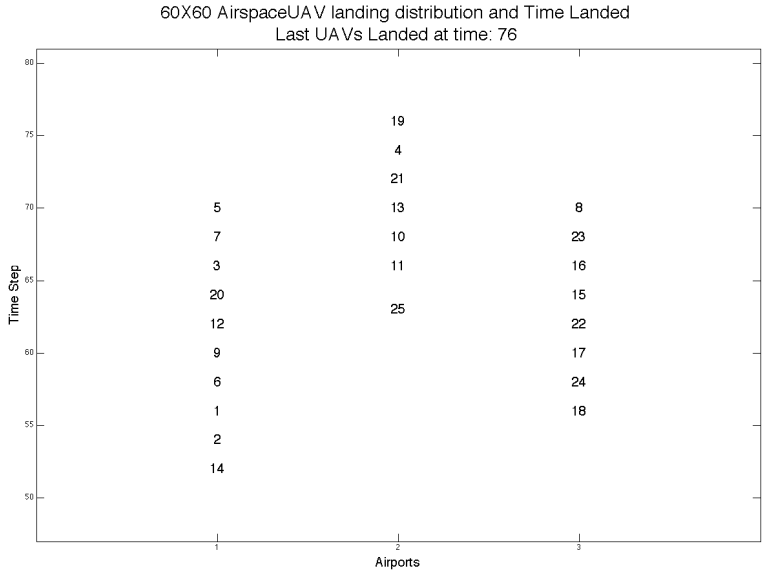


Figure 3.17: Depiction of landing sequence of UAVs in 3 Airports using the Organic Mask Approach. Notice how Airport 1 has only 6 UAV land there, where the other 2 landing zones have 9. Each landing zone is supposed to have at least 7 per. This is an example of the organic mask algorithm relaxing the "required" constraint.

Based on the exploration provided in this thesis, the multiple convex hull approach is the best choice for further investigation into larger airspace examples and comparison with additional simulation models. The multiple convex hull approach consistently meets all constraints and uses less computational power than the other examined approaches across all scenarios. This dominant approach can be seen in the following data plots, which identify the results and performance of the presented algorithms for the number of nodes, number of edges, execution runtime, amount of memory required, the minimal cost function value obtained, and the mission performance metric of total time in system (TOS) of the swarm of UAVs.

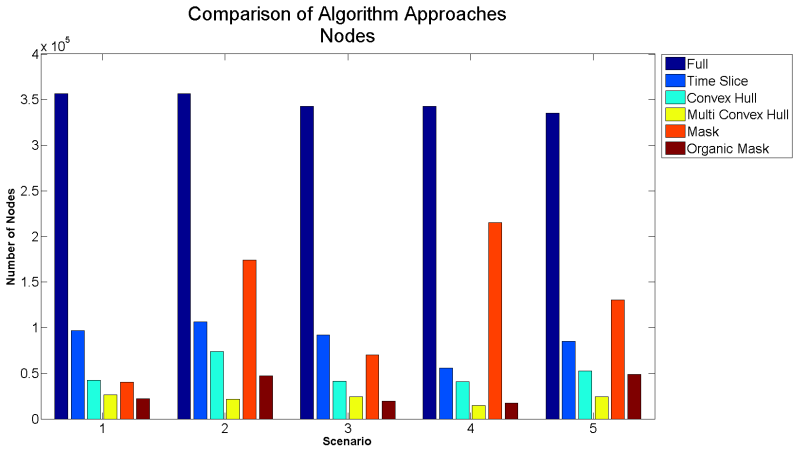


Figure 3.18: This graph shows the significant node reduction capability of the algorithms with respect to the Full airspace model, with the organic mask and multi-convex hull algorithms reducing the number of nodes most significantly.

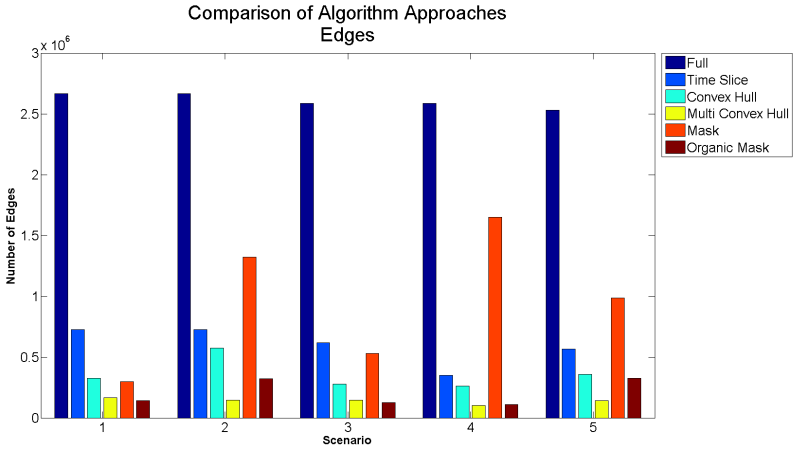


Figure 3.19: This graph similarly shows the reduction of the number of edges in the network, with the organic mask and multi-convex hull algorithms reducing the number of edges most significantly.

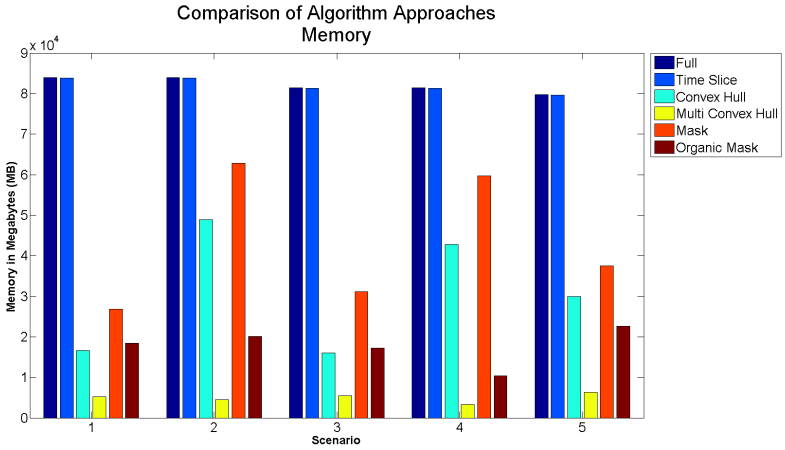


Figure 3.20: Comparison of the amount of memory used by GAMS/CPLEX to solve each algorithm, as measured in megabytes(Mb) is depicted in this chart. The multi-convex hull provides the lowest memory requirement, whereas the organic mask algorithm differs with its increased requirement (though still less than others). Of interest is that the time-slice and full algorithms have the same memory requirement, since all elements of the airspace must be evaluated for these two approaches.

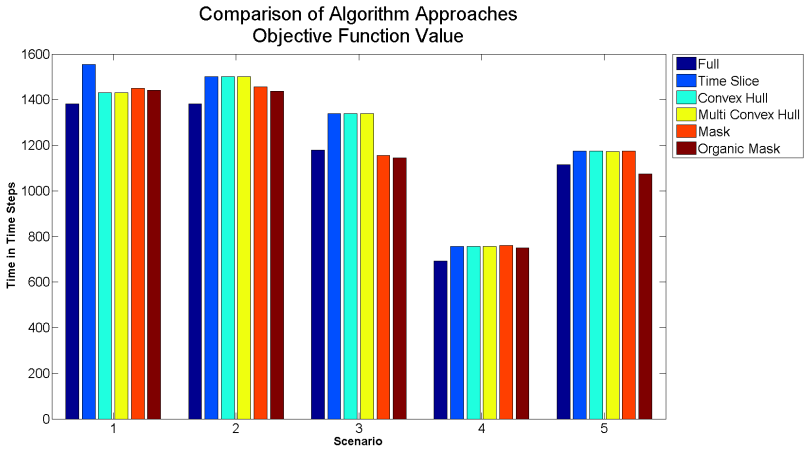


Figure 3.21: Comparison of the minimized objective function value for each algorithm is shown in this graph, and is measured in number of time steps. The full airspace algorithm provides the optimal benchmark solution, where all other algorithms yield sub-optimal values in all cases. The organic mask algorithm violates the constraint (Equation 2.4) in scenarios three and five, which is seen by its lower objective function values for those scenarios.

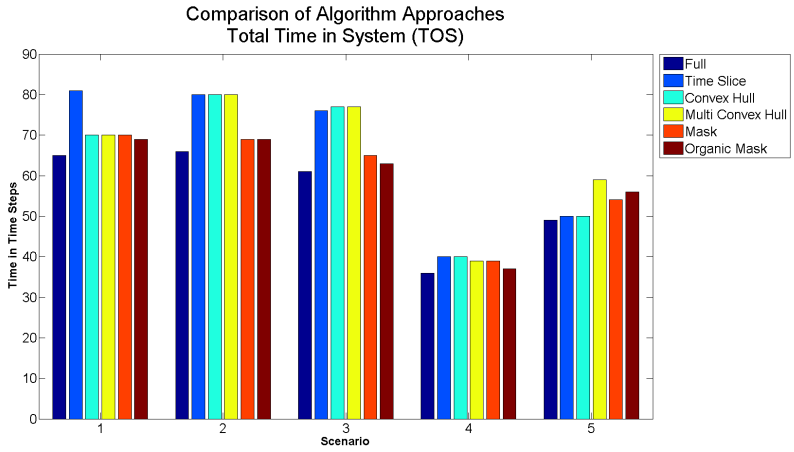


Figure 3.22: The mission performance metric, of interest to the decision maker, is the total time in system (TOS) of the UAVs in the airspace, i.e., the time at which the last UAV exits the airspace. For each algorithm, this time is recorded and contrasted against the full algorithm, which is presumed to yield the minimum TOS value of all investigated airspace models.

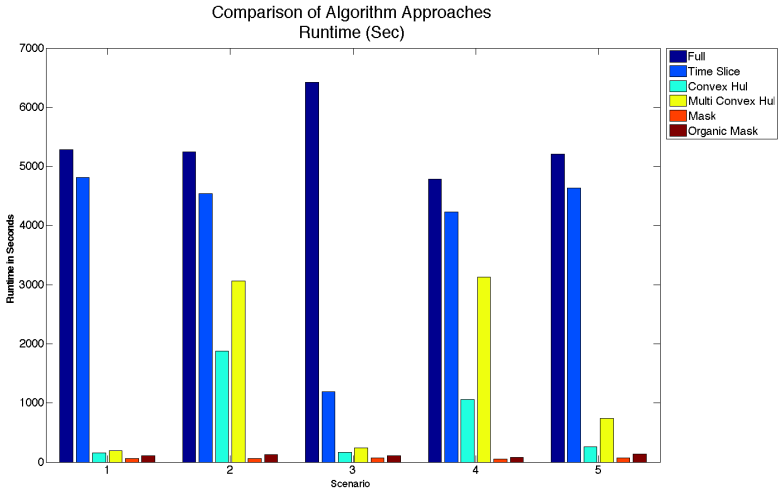
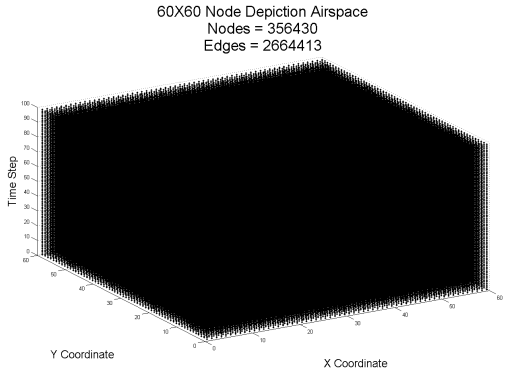
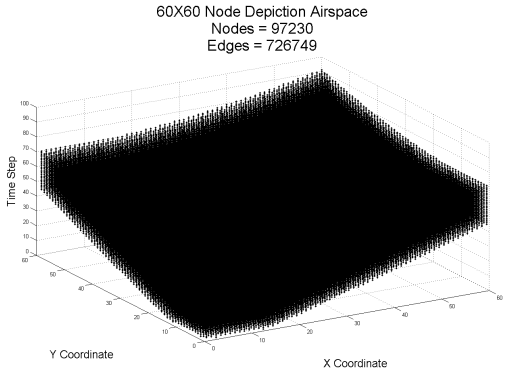


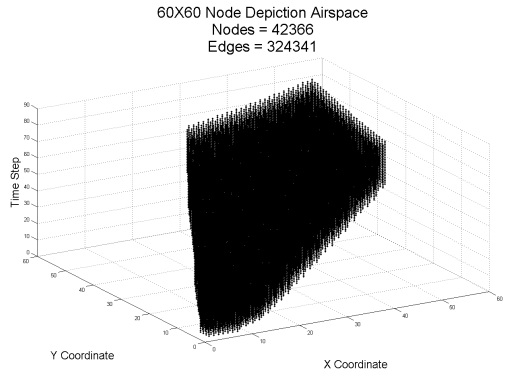
Figure 3.23: The execution time, as measured in seconds, highlights the variable amounts of time to find a solution for the different algorithms in GAMS. Once again, we find the multi-convex hull and organic mask algorithms to (nearly identically) find the solutions fastest across all scenarios.



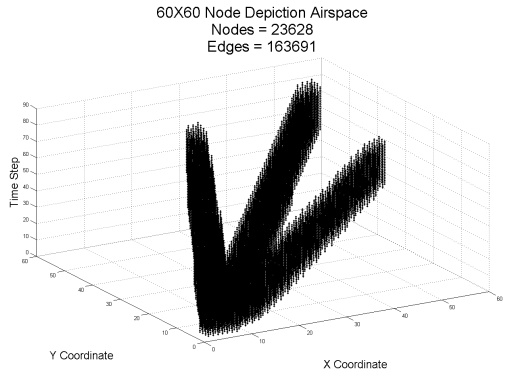
(a) Full Airspace



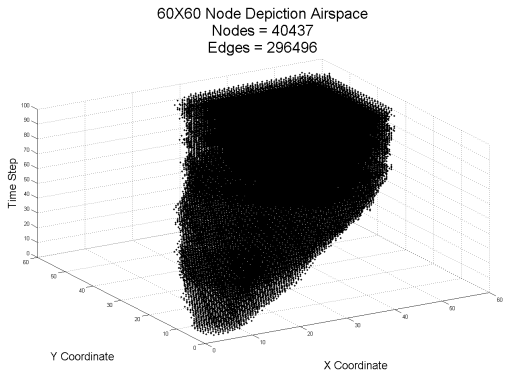
(b) Time-sliced



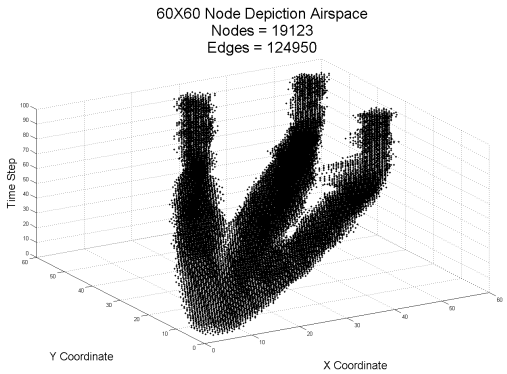
(c) Convex hull



(d) Multi-convex hull



(e) Agent-based Mask



(f) Organic Mask

Figure 3.24: Illustrations of the proposed networks for a 60×60 airspace, as constructed by the proposed network construction algorithms. Each node in the network is represented as a single black "·". As we use the different algorithms, we see the drastic difference in the number of nodes representing the airspace network, thereby impacting the computational feasibility of the optimization problem.

CHAPTER 4:
Model Comparison

4.1 Introduction

In this chapter we develop and discuss an Agent Based Model (ABM) as an approximation approach to the optimization formulation presented previously in Chapters 2 and 3. We choose to use an ABM because of its reflection of a distributed implementation, potentially relevant to physical operational deployments of UAV systems. The distributed nature of computation arises in the sense that each agent performs its trajectory generation individually. Each agent acts in regards to its own awareness (e.g., from sensor data) of its local environment without requiring communication with other agents or a centralized control node. This chapter begins with a more detailed description of each agent and the collective ABM model.

The key objective of this study is to examine the value of distributed heuristic computation approaches, whether for the purposes of parallel simulation experiments or for operational implementations across many units. We note here, however, that the dichotomy between centralized and decentralized or distributed formulations does not necessarily exactly correlate to the complexity or simplicity of the individual agents, respectively. For example, in the case of UAV swarms, while the centralized solution may require substantial computation and communication resources to execute in order to coordinate the UAV swarm's route planning, each individual UAV may, in practice, be a simple drone only capable of following navigation waypoints, entrusting the centralized node with significant operational oversight. In contrast, though the agent-based model presented in this chapter may offer computational advantages through parallel processing and simpler rule-based algorithms, the individual UAV may now require greater levels of autonomy for local perception of neighboring units, advanced collision avoidance capabilities, and real-time control algorithms to execute the desired flight patterns. Exploration of these often-confounded relationships is one of the contributions of this work.

4.2 Agent-Based Model for UAV Swarms

An ABM is a simulation in which each entity or agent in the simulation individually processes its own surroundings. The agent then acts upon this information and causes a subsequent reaction in the other agents in the simulation. Each agent reacts to the environment (including its neighboring agents) using a set of simple interaction rules. Through these in-

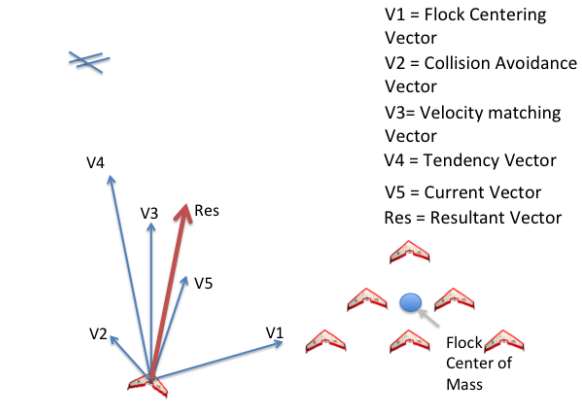
teraction rules, researchers have been able to observe complex emergent behavior and gain understanding about the world. Examples of these emergent behaviors include construction of complex structures (e.g., hives, termite mounds), trends in economic market systems, and interactions of atomic particles [16]. Of particular interest in this present study is the model of flocking described in the sequel.

4.2.1 Boid Flocking Heuristics

In 1987 Craig Reynolds published a paper on the flocking characteristics of birds and other flock animals [1]. Reynolds' paper specifically aimed to mimic this flocking behavior for computer graphics and design, but is applicable to our research for UAV swarms as well as significant other areas of research. Reynolds outlines three basic characteristic "rules" that can be used by computer agents to mimic flocking behavior in a generic creature that he called a "boid," which include:

- **Flock Centering**: A general affinity for like creatures to be close to one another. More specifically it is the desire for agents to move toward the center of mass of the other agents, called the *perceived center*.
- **Collision Avoidance**: This is the opposite of flock centering, where there is a repulsive force causing the boids to want to be apart from each other. Combined with flock centering these two forces cause a push-pull effect that cause agents to be close to one another without colliding.
- **Velocity Matching**: This force causes the boids to want to move in the same direction. Since velocity is a vector, consisting of direction and speed, the boids having the same vector will have the same direction and speed [1]. Combining this with the previous two rules, the flock is now able to move in the same direction in close proximity without colliding.

With these general rules, we can create complex flocking behavior to simulate a variety of situations. From these basic concepts, for the work presented in this thesis, we augment these three forces to include an additional *tendency* force that "pulls" the flock toward a goal, such as a designated landing zone. Figure 4.1 depicts the calculation of these four vectors and the resultant vector, which guides the motion of each agent at each time step. Such calculations are performed individually based on local information, thereby enabling distributed implementations of this agent-based model in practice.



$$Res = v1 + v2 + v3 + v4 + v5$$

Figure 4.1: Graphical depiction of flocking with vector addition described in [1] and implemented in the proposed agent-based model. We see how the addition of several vectors act upon the individual agent to determine its path in the next time step.

4.2.2 Proposed Flocking Algorithm

As Reynolds outlines [1], we can create this flocking behavior by the addition of appropriate vectors. A vector for each of the behaviors that we desire (i.e., attraction, repulsion, alignment, and tendency) is created and vector summed for each agent. The resulting vector directs each agent where to move in the next time step. Once the resultant vector is realized, it is normalized so that the UAVs all travel at the same speed per time step, analogous to the constant speed assumption made in the network-based model representation in the previous chapters.

Parker created a short tutorial online [26], providing an overview of Reynolds' boids in the form of pseudo-code, and Pillai [27] from the University of Michigan at Ann Arbor translated this pseudo-code into Matlab, and was generous enough to allow us to use it for our research. This code was used as a starting point for the implementation of the ABM presented herein. Some major modifications include the ability to specify multiple goal locations (i.e., multiple landing zones), as well as the addition of flags for regulating (e.g., appropriate intervals between landings) and registering (e.g., counting and dismissing agents) the arrival of UAV agents. Additional changes to the weights of the vectors were made as the result of trial and error to produce consistent and desired system behavior. The modified algorithm for updating each agent's path at each time step is outlined in Algorithm 4.1. The specific executable code implementing this model can be accessed at: http://faculty.nps.edu/thchung (under software).

Algorithm 4.1 Algorithm for agent movement vector calculations

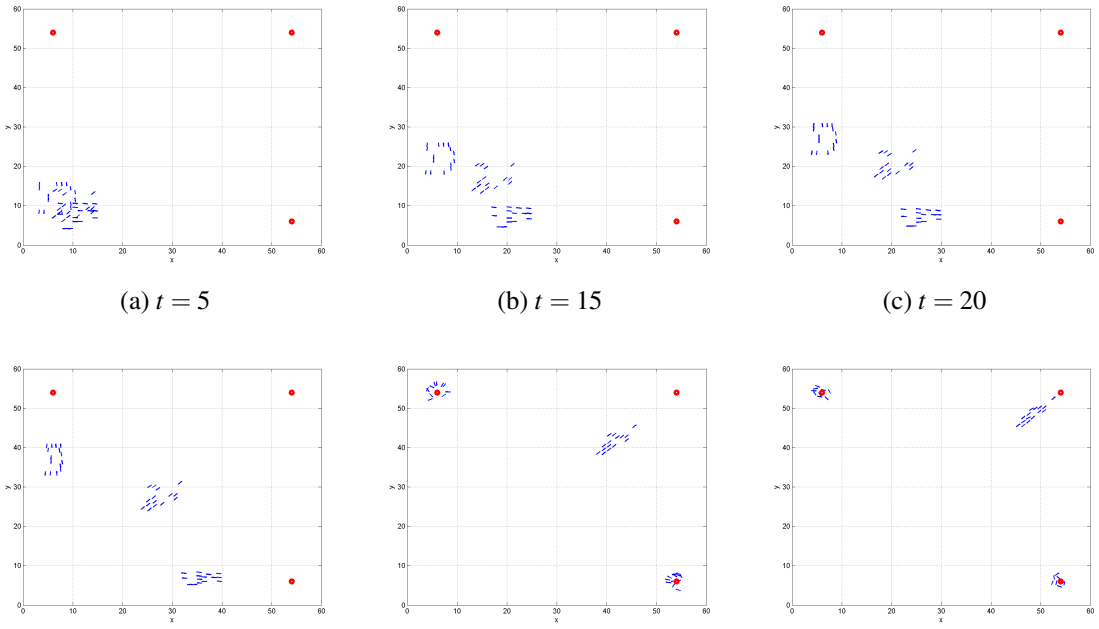
for all $t \in T$ **do**
  for all Boids **do**
    $drive = \max(3, \log(t^2))$ {Weight for agents' attraction to the landing zone}
    $v_1 =$ Flock Centering Vector
    $v_2 =$ Collision Avoidance Vector
    $v_3 =$ Velocity Matching Vector
    $v_4 =$ Tendency Vector
    $v_5 =$ Current Velocity Vector
    $v_{\text{Res}} = (v_1 + v_2 + (3\ v_3) + drive \cdot v_4) + v_5$
    $CurrentPosition \leftarrow CurrentPosition + v_{\text{Res}}$
  end for
end for

Figure 4.2 provides a sequence of snapshots over the evolution of the mission as applied to the baseline scenario (Scenario One). As before in the network-based model formulation, the UAV swarm is initially located in the southwest corner of the airspace, with three destination airports in the remaining three corners. As the scenario progresses, the UAV agents separate by assigned landing zone and transit towards them using the flocking agent-based model algorithm proposed in this work. The final panel depicts the arrival of the UAVs at their respective landing zones, at which point the mission performance metric of total time in system (TOS) is recorded.

4.3 Agent-based flocking model results

The ABM was applied to each of the five scenario discussed in Chapter 3, with the same inputs as the network-based model. These simulation experiments provide likely trajectories for the UAVs as they transit the airspace, recognizing the nondeterministic nature of the simulations. The resulting performances can then be compared and contrasted to the results from our network-based models. It is important to realize that these agent-based models do not produce a complete flight control path for the UAVs as we did in the network-based models; rather, the rule-based control laws enable the computation of local maneuvers to be executed by each UAV. With these agent-based model rules, we are comparing an environment where the UAVs have no centralized controller; instead, each instance of the simulation can be thought of as an actual nondeterministic flight of the UAVs, and is treated as a sample path in a distribution of swarm trajectories. This concept is addressed later in this section.

As before, the following simulation experiment parameters are held constant through the



(a) $t = 5$ (b) $t = 15$ (c) $t = 20$

(d) $t = 30$ (e) $t = 50$ (f) $t = 60$

Figure 4.2: Select frames from our ABM over the evolution of the simulation. The red circles depict the landing zones, the blue lines are the individual agents. We can see how the agents separate into their individual groups as they fly to their respective landing zones. We can also see the interaction of the separate vectors, such that the agents are capable of flying in close proximity to each other without collision and in an apparently coordinated manner.

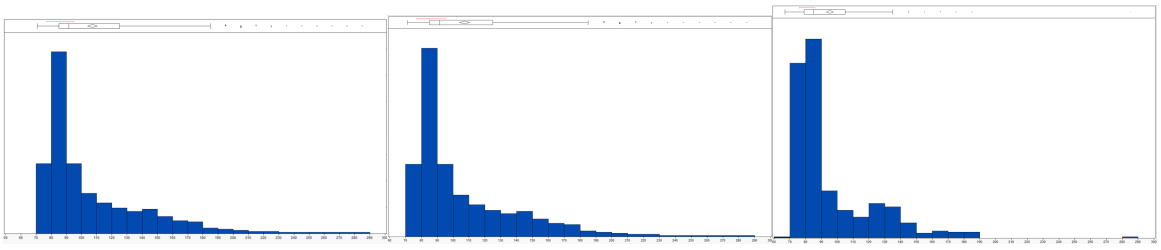
replications, in order to highlight the differences between the agent-based model approach and the network-based model formulations, with 500 replications of the simulation experiments to provide statistical measures:

- 25 UAVs
- 60×60 airspace
- Identical initial positions for all UAVs
- Fixed landing zone locations
- 500 replications per scenario

Table 4.1 summarizes the results of the simulation experiments conducted for all five scenarios, including the statistics of the mission performance, that is, the total time in system (TOS), of the agent-based flocking approach.

Table 4.1: Summary simulation results for each of the five scenarios with 25 UAVs operating in a 60×60 airspace. The summary statistics is for 500 replications of the simulation experiment, including the minimum and maximum values of the total time in system (TOS) of the UAV swarm, as well as the median, and mean times, their standard deviation, and the total computation runtime.

Scenario	Min	Max	Mean	Median	SD
1	71	285	107.13	92	35.424
2	69	110	85.18	86	5.511
3	67	285	95.41	85	26.099
4	41	275	84.77	75	36.930
5	50	165	73.39	65	23.075



(a) Scenario One Data (b) Scenario Two Data (c) Scenario Three Data

(d) Scenario Four Data (e) Scenario Five Data

Figure 4.3: Frequency histograms of the simulation data, with the horizontal axis representing total time in system, obtained by running the agent based model through each scenario 500 replications. The histograms reflect a distribution with long tails and some extreme outliers.

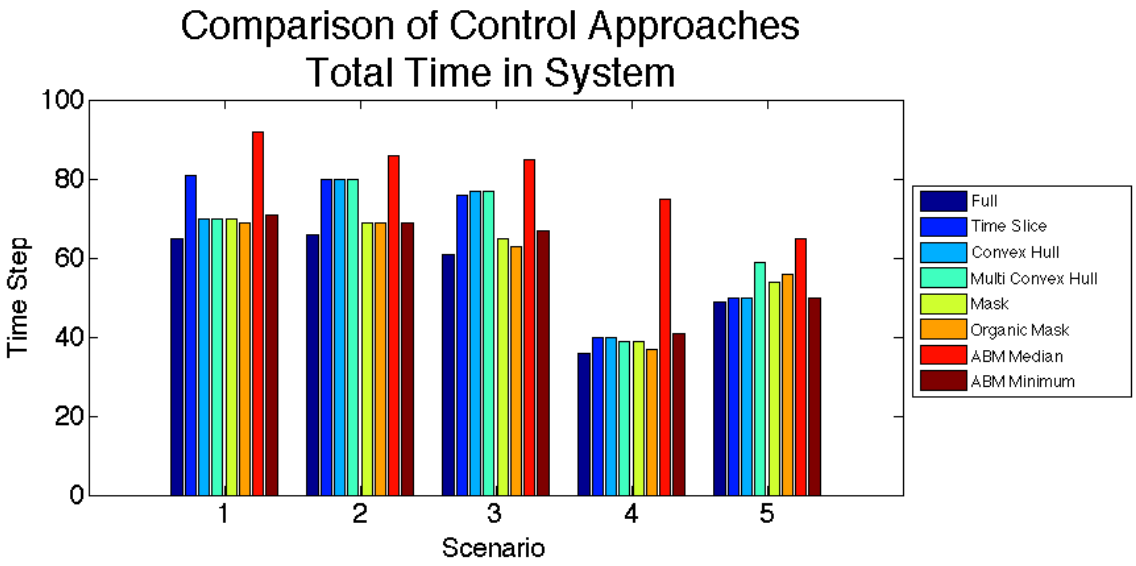
We can see from the histograms of the data in Figure 4.3 that the distributions possess long tails with a few outliers that pull their respective means toward these values. For this reason, we use the median as a measure of central tendency for representing our results. In this manner, the median is more robust and less sensitive to outlying values, and provide us with a better estimate of the expected performance of the UAVs as they move through the airspace.

Of note is that the variance of landing times in the simulation model is larger than anticipated, which is likely explained by the interaction between the agents. Agents in the

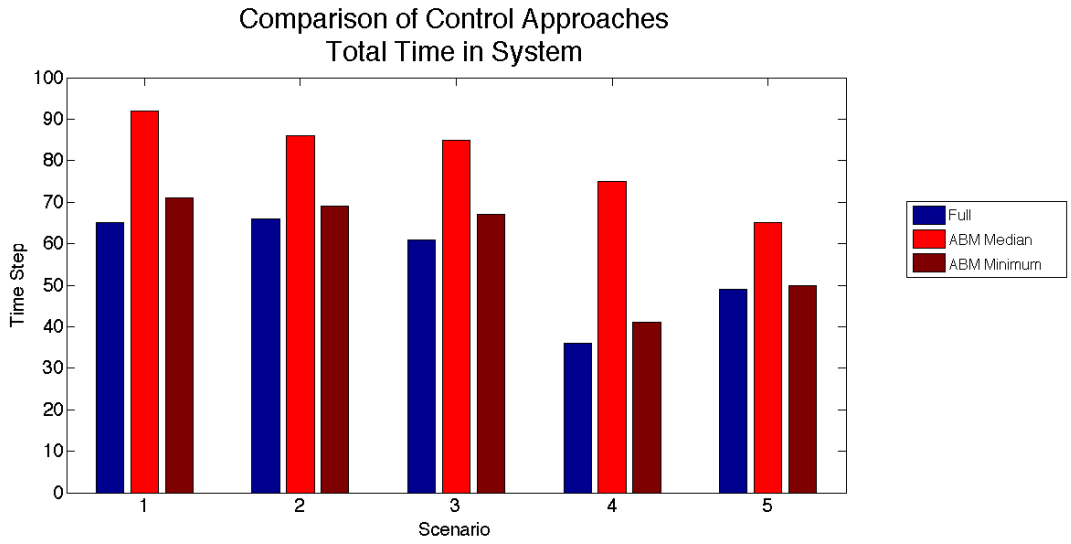
vicinity of the landing zone repel each other, frequently preventing these agents from landing. In these instances, the agents actually exhibit a holding pattern (that is, an equilibrium) behavior around the landing zones, as the agents are unable to turn fast enough to go directly into the landing zone. This mutual "blocking" takes a relatively long time for the agents to resolve, causing a large range in landing time values. This problem is overcome by introducing two elements, including a time-varying attraction factor to the landing zone, where as time increases so does the attraction to the LZ. The other is a periodic negation of the "tendency" vector, that is, every ten time steps a repulsive force is introduced (instead of the usual goal-attractive force) to push the agents away from the landing zone. This momentary perturbation disrupts the balance of forces and enables agents to once again attempt to sequence their landing. As the goal of this study was to introduce the application of agent-based flocking models, these artifacts are likely due to the simple interaction rules presented. Future explorations can enrich these interaction rules such that the variance exhibited by the present study is minimized.

4.4 Comparison between flocking simulation and network-based optimization algorithms

The value of the network-based approaches outlined in the previous chapters was the ability to provide a measure of optimality of solutions. In that manner, the agent-based simulation model presented in this chapter is consistently less efficient in the total time in system than any of the algorithms explored previously for all investigated scenarios, as seen in Figure 4.4. This discrepancy can be partly explained by two factors. First, the network-based model is in discrete space and transits to diagonally adjacent nodes does not incur additional cost as those laterally adjacent, e.g., distance cost is measured by the 1-norm. The simulation model, being in continuous space using the Euclidean distance norm, does apply this penalty. This difference could account for a difference in performance of approximately eight time steps in any of the scenarios. With this handicap in mind, the minimum time in system for the flocking algorithm approaches the performance nearly equivalent to the full network-based model's optimal value. Second, the simulation model more accurately accounts for flight dynamics than does the network-based model, i.e., the agent in the simulation model cannot turn 180 degrees in one time step where the network-based model does not have that limitation (see Figure 2.3).



(a) All Control Approaches



(b) Full Algorithm and Agent Based Model Approaches

Figure 4.4: Comparison of all scenarios with performances for network-based optimization and agent-based flocking algorithms. Simulation median is consistently higher than the optimal values obtained with the network-based approaches.

Nevertheless, even accounting for these mitigating factors, we still observe differences in the time to land the swarm. In Table 4.2, the distributions of the flocking algorithm simulation results for each scenario are compared against the values from the network-based optimization formulation using statistical tests, including the standard t-test (although the results also hold using a Wilson-Cox nonparametric test), which highlight that their differences

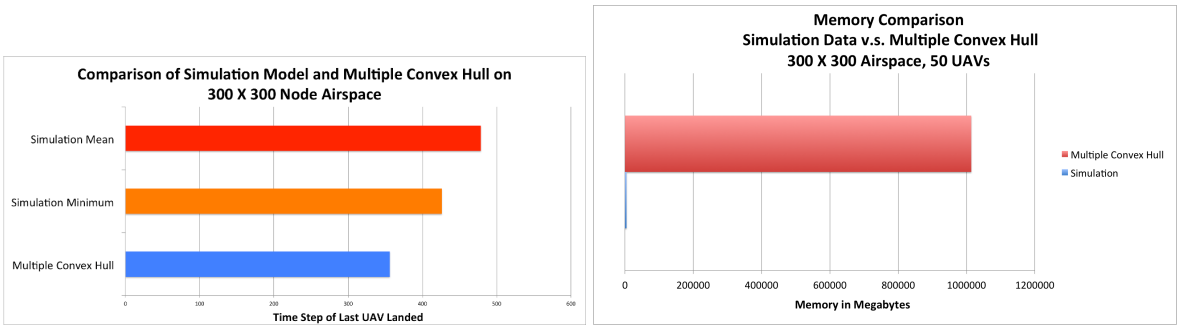
are statistically significant. Interestingly, the minimum values are equal to or approaching the optimal value obtained by our network-based model. This nearly optimal performance shows that, even with simple interaction rules for governing UAV agents, there is significant potential for the distributed flocking approach presented in this thesis.

Table 4.2: Comparison table of the flocking algorithm simulation results against the network-based optimization models. The network-based model produced results that are statistically significantly better than the simulation model, as evidenced by the reported p-values using a t-test being less than 0.0001.

Method	**Value**	**Scenarios**				
		1	**2**	**3**	**4**	**5**
Simulation	μ	107.134	85.18	95.41	84.41	73.392
	σ	35.4241	5.5111	26.099	36.9	37.4899
Full	TOS	65	66	61	36	49
	p-value	$< .0001$	$< .0001$	$< .0001$	$< .0001$	$< .0001$
Time-sliced	TOS	81	80	76	40	50
	p-value	$< .0001$	$< .0001$	$< .0001$	$< .0001$	$< .0001$
Convex Hull	TOS	70	80	77	40	50
	p-value	$< .0001$	$< .0001$	$< .0001$	$< .0001$	$< .0001$
Multi Convex Hull	TOS	70	80	77	39	59
	p-value	$< .0001$	$< .0001$	$< .0001$	$< .0001$	$< .0001$
Agent-based Masks	TOS	70	69	65	39	54
	p-value	$< .0001$	$< .0001$	$< .0001$	$< .0001$	$< .0001$
Organic Masks	TOS	69	69	63	37	56
	p-value	$< .0001$	$< .0001$	$< .0001$	$< .0001$	$< .0001$

4.4.1 Flocking model vs. multiple convex hull network-based optimization model

The results from the previous investigation of the network-based formulations highlights the multiple convex hull approach as the recommended choice for constructing the airspace network. In order to more thoroughly explore the (dis)advantages of the agent-based model approach, we conduct a focused study of Scenario One using a significantly larger airspace of 300×300 with a full complement of 50 UAVs in the swarm, allowing for five time step intervals between landing UAVs.



(a) Landing Times

(b) Memory

Figure 4.5: Comparison of simulation model and the multiple convex hull approach, with (a) the landing time comparison, in time to land the swarm, and (b) the memory in megabytes (MB) that it took the computer to produce the above answers. We can see that the multiple convex hull approach is significantly more efficient than the distributed approach in landing the swarm, but at the computational cost of over 1 Terabyte of computational power.

We can see in Figure 4.5(a) that the multiple convex hull approach is considerably more efficient (in terms of UAV swarm time in system) than the agent based flocking model for this specific airspace and scenario. In other words, the network-based optimization approach is nearly 20% lower in TOS as compared to the agent-based model's minimum value, and exhibits a nearly 35% advantage over the agent-based model's median performance.

However, the memory comparison provides a very different picture. As seen in Figure 4.5(b), we observe that the computational cost of the agent-based model scales linearly as the problem gets larger. However, the network-based optimization model as formulated scales exponentially, becoming more and more difficult to solve larger problems. The flocking algorithm (which offers a distributed implementation that would make the computational cost savings more extensive) requires only a fraction of the computation power (in this example, a difference of over 1 Terabyte) when compared to the centralized formulation.

This example shows the power of the distributed solution when we increase the breath and depth of a problem. A distributed, agent-based approach is able to better handle a larger more complex (and potentially more operationally relevant) environment than the centralized approach. Though this trade off may result in 20-40% loss in the time it takes to land the UAV swarm, awareness and knowledge of this trade off can inform areas for improving such distributed algorithm approaches to increase their efficiency.

4.5 Discussion

This chapter investigated an agent-based model using flocking algorithms that presents a distributed alternative to the centralized, computationally expensive approaches presented in

previous chapters. The agent-based heuristic approach is more computationally feasible for larger problem and easier to deploy for expeditionary environments, where computational power and/or the computation time may not readily available. However, we see that the distributed solutions may be far from the optimal solution, and as such may be less useful to the decision-maker.

Specifically for this problem of optimizing the landing of UAV swarms, an observation in the flocking algorithm context is that the individual agents do not start to separate and sequence prior to arriving at the landing zone, in contrast with the network-based models, where the agents anticipate their sequence for landing well before they arrive at the LZ. The agent-based model, being a reactive and myopic model, is unable to provide such insight and plan for extended time horizons.

An additional limitation of the agent-based model presented in this chapter is that the agents are arbitrarily and/or manually assigned to one of the airports. This real-time assignment is not the case in the network-based model where optimal assignments are implicitly part of the computation (captured by constraints on required number of UAVs per landing zone). The addition of an assignment algorithm into the simple flocking model could add increased complexity into the model, although one-shot linear assignment implementations can be solved quite efficiently. In the absence of such predefined assignments of landing zones, respecting allocation requirements of agents to different landing zones would be difficult (e.g., agents would potentially tend to land at the same landing zones due to the Flock Centering behavior)Appropriate tuning of the weights governing the flocking behaviors, e.g., the Flock Centering vectors, may be modified to enable sufficient segregation of the UAV agents into the various landing zones, but is reserved for future study.

Despite these limitations, however, the execution of the agent-based model simulations are computationally inexpensive and highlight their potential significant benefit to exploring these optimization problem contexts. The memory used for every run was within 4 Gb and only required less than nine minutes for the completion of 500 runs! This memory usage is merely an eighth of the available resources, and further highlights the fact that high-performance computing solutions are not always necessary. Further, keeping in mind the fact that the time spent conducting the simulation runs assumed running all the UAVs' flocking behavior computations simultaneously, in actual distributed UAV operations, the computational advantage would be more pronounced. Spare computational capabilities of the UAV could be better utilized for, e.g., sensing and perception subsystems, which may make the agent-based heuristic models attractive when considering the UAV system. The trade off is that in order to create agents that are able to implement a distributed solution, the

agents themselves may become more complex and expensive by requiring such subsystems, whereas the centralized solution requires only simple waypoint execution from the UAV platforms.

Though the results for the agent-based flocking algorithm were far from an optimal solution, they are simply achieved and easily interpreted. The value of the network-based model in this instance is to provide a benchmark against which we can compare our approximation approaches. Recognizing that there are many different ways to produce flocking behaviors, we explored one such construction as presented in this chapter.

CHAPTER 5:
Conclusion and Future Work

5.1 Conclusions

We have been able to produce transit paths for UAV swarms that provide both collision avoidance and efficient routing through airspace. This is done through both the formulation and use of a centralized network-based model as well as a distributed, agent-based heuristic model for flocking. Both models provide deconfliction and routing through the airspace to get the UAVs to one or more landing zones in a relatively efficient manner. The measure of mission performance in these types of operations is the total time in system (TOS), representing the time when the last aircraft lands, which both models seek to minimize.

First we construct the optimization problem as a single commodity, network flow formulation, with side constraints which accounts for the number of UAVs, deconfliction of airspace, and the ultimate and timely arrival of UAVs at landing zones. Given the computational expense of constructing an operationally relevant network, we investigate and propose several algorithmic methods to reduce the number of elements (nodes and edges) in the network. These methods leverage notions of reachable sets and easily computed convex hulls to dramatically decrease the problem space. Further, we explore the use of agent-based simulation as another means for reducing the number of network nodes and edges. Computational routines in GAMS/CPLEX are implemented and applied to find optimal solutions for these respective approaches, and thorough comparison studies highlight a recommended approach of using multiple convex hulls to cheaply inform the network construction.

To enhance the relevance and contribution of this work, we also constructed a purely agent-based simulation approach, using well-known flocking algorithms to represent simple agent interaction rules to govern UAV behaviors. We found that the agent-based model implemented for this study are limited in producing (near) optimal solutions, given a number of reasons including the myopic nature of the agent behaviors. However, the benefit of the investigated flocking algorithm was to highlight the extreme computational advantages in producing approximate (though occasionally severely sub-optimal) solutions to the UAV swarm landing problem.

The advantage of the centralized network-based model is that, though computationally expensive and solved offline, once done so, the computed UAV routes are simple to execute since the paths can be sent to the individual UAVs as waypoints for them to follow. This

would make a good solution for disposable UAV systems, where the expensive component is the computational hardware of the operational planning system (e.g., at the base station) and the UAVs themselves are cheap. Such a capability is readily available, i.e., it requires that the UAV have a simple auto pilot and interface software and does not require onboard sensing nor coordination among assets. The disadvantage of this approach is that currently its application is not computationally feasible for large, operationally relevant scenarios (i.e., involves a prohibitive number of node/edge interactions, and takes hours or longer to solve).

The distributed, agent-based approach is considerably more adaptable and less computationally prohibitive than the centralized solution. This increased flexibility and responsiveness may, however, come at the cost of increasing the complexity of the UAV agents themselves. In other words, the flocking algorithm now relies on the individual UAV's ability to observe its environment, orient to that environment, decide and act accordingly (i.e., perform the OODA loop). The UAV is now responsible for this OODA loop, requiring greater autonomy than is presently fielded in operational systems. The other trade off is that the UAV swarm, on average, will achieve sub-optimal performance using the agent-based approach relative to the centralized network-based model, as the UAVs will be reactive (rather than deliberative), that is, will have little or no considerations for overcoming local minima to achieve optimality.

The network-based optimization formulation for this problem of landing large numbers of UAVs at multiple landing zones is not computationally solvable without the use of high power computing resources. Though the unconstrained airspace, or full, problem is able to produce the best solutions (since there are no restrictions on possible UAV routes), its use requires the trade off of only applying to smaller airspace sizes and less relevant problems. The use of agent-based models and other techniques enable the use of the network-based formulation for larger problems by restricting the problem and "cutting" the airspace into only a portion of what was originally considered. These techniques used to restrict the problem make the optimization more computationally attractive, but at the cost of a sub-optimal solution. Thus combining the use of agent-based models or simulation with optimization models is able to make this problem more tractable and solutions readily available for further use. This synergy is an example where the combination of two disciplines in operations research can produce better analysis and results, solving problems that previously may not have been possible separately.

5.2 Discussion

The melding of both simulation and optimization can produce results that are both tractable and more consistently closer to optimal than simulation alone. Techniques such as agent-

based simulation can be used in traditional optimization models to gain insight into systems and to give the modeler a localized area for finding the solution. In this thesis, we use agent based models to make otherwise computationally intractable problems tractable, while still maintaining a near optimal solution.

Similarly, formulation as an optimization problem gives the simulation modeler analytic bounds on the behavior that is desired, and a benchmark for the measure of performance for a simulation. This benchmark is important for transitioning or extrapolating insights attained from simulation to real and operational systems

It is important to realize that this research is the ground floor of a much larger effort. Some of the techniques developed can be used, not only for the paths of individual UAVs, but for providing a basis for the use of heuristic and exact optimization solution methodologies together. In this example of UAV swarms, a network-based model can be formulated and used for routing and target assignment of squads and sub-swarms, and heuristic models can used for the micro-level control of collision avoidance and flocking, thereby providing a more defined balance between distributed and centralized control.

This thesis further provides important insight into the level of command and control that is required in swarming systems. In the agent-based model we can see that the agents' collective performance is relatively diminished, since the agents are not provided with a "big picture plan" on how to achieve there goal; they act purely on local and immediate information. The use of agents in the network-based model produced results that were sub-optimal but closer to the optimal solution than the purely distributed solution. As such, network optimization models should be used for higher level optimization tasks, for example, target or landing zone assignment, general routing, and large scale airspace deconfliction, leaving the low level control for collision avoidance and flocking to the individual UAV. The most efficient use of resources, both computationally and temporally, may be to have a combination of centralized and distributed optimization and control approaches, perhaps similar to current military hierarchy and command structure in place today.

5.3 Avenues for Future Research

There are several avenues for future research. The network-based model can be expanded and explored in several ways. First, the upper values of the edges can be extended to account for bounds greater than one. By doing so, the network can be expanded to serve as an assignment and macro-routing problem, where the nodes are farther apart and the edges allow squads of $3-50$ UAVs to traverse them, allowing for higher level planning that can provide rough macro control of the UAVs.The network could also be used to assign UAVs

to a specific landing time or window.Another area of expansion is the implementation of a variable time step or variable edge capacities, such that the network behaves differently depending on where in the transit the UAVs are. For example, if the UAVs are en route, then the upper limit on the edge could be 15-20 UAVs, where as they get closer to the landing zones, the time step could become shorter and capacity limits on each edge could tighten.

The network-based model can also be expanded to use the actual flocking rules in the agent-based model to cut the nodes and make the mask. These rules are more complex than the simple particle rules that were used in our agent-based mask algorithms. Thus, the flocking simulation could be run 10 -50 times over the same set of masks using flocking rules, thereby capturing a better representation of the actual nodes that would most likely be used. Also, if a future implementation could record the adjacencies produced by this simulation, then we could "grow" the network vice cutting the original (full airspace) network, possibly providing a better solution to the problem.

The agent-based model can be improved in many areas, including the addition of better or aircraft-relevant flocking rules to improve the agent interactions, perhaps by optimizing the weights on the flocking vectors for more consistent and predictable behavior.

Finally, an area of promising future research is in the explicit integration of the centralized network-based model and the agent-based models together. The network-based model could be used to provide coarse guidance and control, while having the agents provide collision avoidance and flocking behavior. This hybrid approach would involve the passing of coordinates provided from the network-based model to an agent-based simulation or to actual UAVs through a centralized ground controller, which could be directly translated into currently operational UAV systems.

REFERENCES

[1] C. W. Reynolds, "Flocks, herds and schools: A distributed behavioral model," *ACM SIGGRAPH Computer Graphics*, vol. 21, no. 4, pp. 25–34, Aug. 1987. [Online]. Available: http://portal.acm.org/citation.cfm?doid=37402.37406

[2] J. Arquilla, "Swarming and the Future of Conflict," DTIC Document, Tech. Rep., 2000. [Online]. Available: http://oai.dtic.mil/oai/oai?verb=getRecord\&metadataPrefix=html\&identifier=ADA384989

[3] S. Edwards, "Swarming and the Future of Warfare," RAND Corporation, Santa Monica, CA 90407-2138, Tech. Rep., 2005. [Online]. Available: http://oai.dtic.mil/oai/oai?verb=getRecord\&metadataPrefix=html\&identifier=ADA434577

[4] B. Clough, "UAV Swarming? So What are Those Swarms, What are the Implications, and How Do We Handle Them?" DTIC Document, Tech. Rep., 2002. [Online]. Available: http://oai.dtic.mil/oai/oai?verb=getRecord\&metadataPrefix=html\&identifier=ADA405548

[5] S. Griffith, "The Art of War," *Trans.)(London, Oxford University Press, 1963) p*, 1994. [Online]. Available: http://www.stanford.edu/class/polisci211z/1.1/SunTzu.pdf

[6] W. Hennigan, "New drone has no pilot anywhere, so who's accountable?" 2012.

[7] W. D. Shannon, "Future Trends in Unmanned Aviation and Weapons," 2012.

[8] M. D. Peterson, D. J. Bertsimas, and a. R. Odoni, "Decomposition Algorithms for Analyzing Transient Phenomena in Multiclass Queueing Networks in Air Transportation," *Operations Research*, vol. 43, no. 6, pp. 995–1011, Nov. 1995.

[9] M. A. Bolender and G. L. Slater, "Evaluation of scheduling methods for multiple runways," *journal of aircraft*, vol. 37, no. 3, pp. 410–416, Oct. 2000. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/21996520

[10] A. P. Saraf and G. L. Slater, "Optimal Dynamic Scheduling of Aircraft Arrivals at Congested Airports," *Journal of Guidance, Control, and Dynamics*, vol. 31, no. 1, pp. 53–65, Jan. 2008.

[11] J. Boesel, "An air traffic simulation model that predicts and prevents excess demand," The MITRE Corporation: Center for Advanced System Development (CAASD), McLean, Virginia 22102, Tech. Rep., 2003. [Online]. Available: http://www.mitre.org/work/tech_papers/tech_papers_03/boesel_ats/boesel_ats.pdf

[12] P. Brooker, "Simple Models for Airport Delays During Transition to a Trajectory-Based Air Traffic System," *Journal of Navigation*, vol. 62, no. 04, p. 555, Oct. 2009.

[13] R. Ahuja, T. Magnanti, and J. Orlin, *Network Flows: Theory, Algorithms, and Applications*, M. Peterson, Ed. New Jersey: Prentice Hall, 1993, vol. 1.

[14] P. Dell'Olmo, "A new hierarchical architecture for Air Traffic Management: Optimisation of airway capacity in a Free Flight scenario," *European Journal of Operational Research*, vol. 144, no. 1, pp. 179–193, Jan. 2003. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0377221701003940http://www.sciencedirect.com/science/article/pii/S0377221701003940

[15] K. Artiouchine, P. Baptiste, and J. Mattioli, "The K King Problem, an Abstract Model for Computing Aircraft Landing Trajectories: On Modeling a Dynamic Hybrid System with Constraints," *INFORMS Journal on Computing*, vol. 20, no. 2, pp. 222–233, Sep. 2007.

[16] M. Resnick, *Turtles, Termites, and Traffic Jams: Explorations in Massively Parallel Microworlds*. MIT Press, 1997. [Online]. Available: http://books.google.com/books?hl=en\&lr=\&id=K8P1rX8T4kYC\&pgis=1

[17] S. Conway, "An Agent-Based Model for Analyzing Control Policies and the Dynamic Service-Time Performance of a Capacity-Constrained Air Traffic Management Facility." ICAS, 2006, pp. 1–8.

[18] R. Olfati-Saber and R. Murray, "Flocking with obstacle avoidance: Cooperation with limited communication in mobile networks," in *Decision and Control, 2003. Proceedings. 42nd IEEE Conference on*, vol. 2. IEEE, 2003, pp. 2022–2028. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1272912http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1272912

[19] R. Olfati-Saber, "Flocking for multi-agent dynamic systems: Algorithms and theory," *Automatic Control, IEEE Transactions on*, vol. 51, no. 3, pp. 401–420, Mar. 2006.

[20] H. Yu, J. Jian, and Y. Shen, "Flocking Control of a Group of Agents Using a Fuzzy-Logic-Based Attractive/Repulsive Function," *Int'l J. of Communications, Network and System Sciences*, vol. 03, no. 06, pp. 569–577, 2010. [Online]. Available: http://www.scirp.org/journal/PaperDownload.aspx?DOI=10.4236/ijcns.2010.36076

[21] G. G. Brown and R. F. Dell, "Formulating integer linear programs: A rogues' gallery," *INFORMS Transactions on Education*, vol. 7, no. 2, pp. 153–159,

January 2007. [Online]. Available: http://oai.dtic.mil/oai/oai?verb=getRecord\&metadataPrefix=html\&identifier=ADA487382

[22] R. Rosenthal, *GAMS - A User's Guide*, GAMS Development Corporation, Washington, DC, USA, January 2012.

[23] Unkown, *CPLEX 12*, GAMS Development Corporation, http://www.gams.com/dd/docs/solvers/cplex.pdf.

[24] S. Boyd and L. Vandenberghe, *Convex Oprtimization*, 1st ed. Cambridge, UK: Cambridge Univ Press, 2004.

[25] M. de Berg, O. Schwarzkopf, M. van Kreveld, and M. Overmars, *Computational Geometry: Algorithms and Applications*, 3rd ed. New York, New York: Springer, 2000.

[26] C. Parker, "Boids Pseudocode," vol. 1, 2001. [Online]. Available: http://www.vergenet.net/~conrad/boids/pseudocode.htmlhttp://scholar.google.com/scholar?hl=en\&btnG=Search\&q=intitle:Boids+Pseudocode\#0

[27] S. Pillai, "conradboid.m," 1996, flocking code in Matlab, borrowed and modified from the authors original.

THIS PAGE INTENTIONALLY LEFT BLANK

Initial Distribution List

1. Defense Technical Information Center
   Ft. Belvoir, Virginia
2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California
3. Marine Corps Representative
   Naval Postgraduate School
   Monterey, California
4. Directory, Training and Education, MCCDC, Code C46
   Quantico, Virginia
5. Marine Corps Tactical System Support Activity (Attn: Operations Officer)
   Camp Pendleton, California
6. Director, Studies and Analysis Division, MCCDC, Code C45
   Quantico, Virginia